Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Vytvoření a použití scénářů WebRTC

Building and Using WebRTC Scenarios

2015 **Bc. Karolína Benešová**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Karolína Benešová**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Vytvoření a použití scénářů WebRTC.
Building and Using WebRTC Scenarios.

Zásady pro vypracování:

Téma diplomová práce je směrováno do oblasti real-time komunikace prostřednictvím webového prohlížeče s využitím WebRTC technologie. Cílem diplomové práce je praktická implementace WebRTC v realizovaných scénářích nasazení WebRTC a porovnání provozních parametrů nasazení WebRTC s dosud používanými samostatnými SIP klienty.

1. Multimédia v prostředí webových aplikací.
2. WebRTC technologie a SIP B2BUA.
3. Praktická realizace různých scénářů nasazení WebRTC.
4. Porovnání WebRTC scénářů s konvenčními SIP relacemi.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

S. Loreto, S. Romano. *Real-Time Communication with WebRTC Peer-to-Peer in the Browser*. O'Reilly Media, 2014, ISBN 978-1-44937-187-6.
A. Johnston, D. Burnett. *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web*. Digital Codex LLC, 2012, ISBN 978-0985978808.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě dne: *6. května 2015*

Bc. Karolína Benešová

.....

Podpis studenta

Poděkování

Ráda bych poděkovala doc. Ing. Miroslavu Vozňákovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

V diplomové práci je pojednáváno o implementaci WebRTC pomocí pobočkové ústředny Asterisk a pomocí SipML5 klienta, který byl vytvořen společností Doubango Telecom. V praktické části je realizováno zprovoznění WebRTC a jeho následné otestování pomocí SipML5 klienta ovládaného přes webové rozhraní. Dále bylo provedeno měření přenosových parametrů technologie WebRTC pomocí paketového analyzátoru Wireshark. Tyto parametry jsou porovnávány s konvenčními SIP relacemi, které využívají pro přenos dat protokol RTP a SRTP. Zpracování a porovnání dat probíhá pomocí software Matlab, ve kterém byly vytvořeny skripty pro analýzu dat. Výsledné naměřené hodnoty jsou následně využity pro nástroj na výpočet celkových nároků na realizaci daného počtu audio a video hovorů.

Klíčová slova

iLBC, iSAC, OPUS, VP8, VP9, H.264, H.265, SIP, WebRTC, RTP, SRTP, UDP

Abstract

This work is about the implementation of WebRTC using a PBX Asterisk and SipML5 client, what was created by Doubango Telecom. In the practical part is realized putting into operation of WebRTC and this is subsequent testing using SipML5 client controlled via a web interface. Subsequently, was made measurement of transmission parameters technology WebRTC using a packet analyzer Wireshark. These parameters are then compared with conventional SIP relationships, which use for the data transfer protocol RTP and SRTP. Processing and comparison of data is done by using Matlab software, in which were created simple scripts for data analysis. The results of measurements are then used to calculate a tool for total demands the implementation of a number of audio and video calls.

Key words

iLBC, iSAC, OPUS, VP8, VP9, H.264, H.265, SIP, WebRTC, RTP, SRTP, UDP

Seznam použitých zkratek

Zkratka	Význam
AVC	Advance Video Coding
B2BUA	Back-to-back User Agent
DTLS	Datagram Transport Layer Security
GOP	Group Of Picture
HEVC	High Efficiency Video Coding
ICE	Interactive Connectivity Establishment
iLBC	internet Low Bitrate Codec
iSAC	internet Speech Audio Codec
JVT	Join Video Team
MPEG	Moving Picture Expert Group
MTI	Mandatory To Implement
PB	Prediction Blocks
PBX	Private Branch eXchange
RIA	Ritch Internet Applications
SIP	Session Iniciation Protocol
STUN	Session Traversal Utilities for NAT
SWF	Shock Wave Flash
TURN	Traversal Using Relays around NAT
UA	User Agent
UAC	User Agent Client
UAS	User Agent Sever
XAML	eXtensible Application Markup Language

Obsah

Úvod

V dnešní době rozšířeného mobilního datového připojení je pro uživatele výhodné využívat volání přes internetovou síť, kde tato komunikace není tarifována a je účtováno pouze datové připojení.

Komunikaci přes internet lze realizovat pomocí pobočkové ústředny Asterisk, kde díky správné konfiguraci PBX Asterisk je možné vytvořit i celou telefonní síť s call centrem a interaktivním hlasovým menu pro zavolání. Nevýhodou je nutnost instalace SIP klienta, který je potřebný pro registraci do sítě dané pobočkové ústředny a pro realizaci hovoru.

V dnešní době chtějí uživatelé strávit minimum času s instalací software a ušetřit kapacitu na disku. Nevýhodou je také rozdílnost nastavení jednotlivých SIP klientů. Každý software má rozdílné nastavení parametrů a proto může mít uživatel při využívání více typů SIP klientů problém. Tyto problémy řeší technologie WebRTC, která využívá PBX Asterisk, ale nastavení klientů je pro uživatele jednodušší. Není třeba instalace klientského software, protože komunikace je realizována mezi webovými prohlížeči. Velký vývojový krok pro tuto technologii udělala společnost Doubango Telecom, která vytvořila online internetové demo SipML5 klienta. Tuto stránku lze využít pro volání mezi klienty bez nutnosti dodatečných instalací. Velkou výhodou je jednoduchost nastavení pro volání a hlavně vždy stejné uživatelské rozhraní. Další výhodou je nezávislost na typu operačního systému a typu zařízení. Uživatel nemusí řešit jednotlivé rozdíly v zařízeních, podmínkou je pouze internetový prohlížeč a přístup do sítě.

Diplomová práce se zabývá problematikou pokročilých služeb v moderní pobočkové ústředně. Praktická část je zaměřena na volně šiřitelné aplikace. Základním nástrojem pro tuto práci bude pobočková ústředna Asterisk a technologie WebRTC. Tuto technologii zprovozním a detailně prostuduji z pohledu nároků na síť. Pro porovnání technologie budou vytvořeny dva další typy SIP relace, které pro přenos datových toků využívají protokolů RTP a SRTP. Bude zkoumán audio i video provoz a naměřená data budou následně vyhodnocena pomocí vytvořených skriptů. Výsledky budou zhodnoceny a bude vytvořen nástroj pro výpočet výsledného zatížení sítě pro dané parametry.

1 Multimédia v prostředí webových aplikací

Pro možnost využití multimédií ve webovém prohlížeči musíme být schopni tyto multimédia přenášet mezi jednotlivými prohlížeči. Pro přenos audio i video souborů jsou definovány přenosové standardy, které určují způsob přenosu multimédia. Některé multimediální obsahy také vyžadují instalaci zásuvných modulů pro jejich podporu, jak je uvedeno dále v kapitole.

1.1 Audio standardy

1.1.1 iLBC

ILBC (internet Low Bitrate Codec) je standard vytvořený firmou GIPS Global IP Solution, která v roce 2011 přešla pod firmu Google a díky tomu je standard vytvářen pomocí SBD licencí. Tento audio standard se využívá především pro VoIP, streamování audia, videokonferenci atd.

Standard umožňuje snížení kvality řeči vzhledem ke ztrátovosti paketů, která je způsobena ztracenými nebo opožděnými IP pakety.

Vzorkovací frekvence je 8kHz. Jsou využívány dvě různé velikosti rámců, 400 bitů pro rámec obsahující 30 ms audia a 304 bitů pro rámec s 20 ms audia. Přenosová rychlost je také rozdílná pro rámec s 30 ms audia je to 13,33 kbit/s a pro rámce s 20 ms je to 15,2 kbit/s. [3]

1.1.2 iSAC

ISAC (internet Speech Audio Codec) je adaptivní širokopásmový standard pro řeč a audio. Standard pracuje s malým časovým zpožděním a je vhodný pro vysoce kvalitní realtime komunikaci. Je také schopen zpracovat audio, které neobsahuje řeč, jako je například hudba, nebo šum v pozadí.

Audio je zpracováváno pomocí 16kHz vzorkovací frekvence (popřípadě 32kHz v případě webRTC) s variabilní přenosovou rychlostí 10 kbit/s až 32 kbit/s (u webRTC až do 52 kbit/s). Každý rámec obsahuje 30 až 60 ms přenášeného zvuku.

Standard má definován dva režimy:

Channel-adaptive – Přenosová rychlost je přizpůsobena, aby odpovídala dostupné šířce pásma, což poskytuje vysokou kvalitu audia. Standardní přenosová rychlost adaptivního módu je 20 kbit/s.

Channel-independent - U tohoto módu je přenosová rychlost poskytnuta již před kódováním.

Standard využívá bezeztrátové kódování, které ještě více snižuje velikost jednotlivých paketů a tím celkově snižuje přenosovou rychlost. [2]

1.1.3 **OPUS**

Je otevřený vysoce univerzální zvukový standard pro přenos hlasu přes internetovou síť. Jedná se o standard IETF RFC 6716. [1]

Opus je určen pro širokou škálu využití. Je možno jej využít pro VoIP, videokonferenci a vzdálené hudební vystoupení. Jednotlivé vlastnosti standardu jsou velmi variabilní:

- Přenosová rychlost od 6 kbit/s až do 510 kbit/s
- Vzorkovací frekvence od 8 kHz do 48 kHz
- Velikost paketů 2,5 ms až 60 ms
- Je určen pro úzkopásmový přenos i pro přenos v rámci celého dostupného pásma
- Podporuje řeč i audio

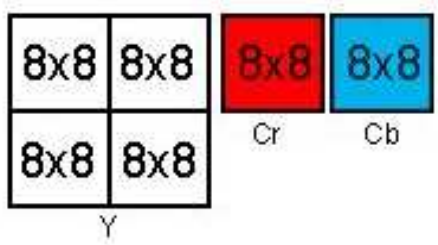
1.2 Video standardy

1.2.1 **H.264**

V roce 2013 byl schválen standard pro kompresi signálu h.264 nebo také MPEG-4 part 10 či MPEG-4 AVC (Advanced Video Coding). Tento standard je navržen především pro systémy s vysokým rozlišením (HD), například pro satelitní digitální vysílání ve vysokém rozlišení HDTV. Úkolem je přenášet obraz ve vyšší kvalitě při nižší přenosové rychlosti (až dvakrát lepší kvalita při stejné velikosti než u MPEG-2), proto se hodí jak pro webové přehrávače s nízkými přenosovými rychlostmi, tak pro přenosná zařízení či nejkvalitnější video.

Tento standard vyvinul tým JVT (Join Video Team), který vznikl spojením skupiny MPEG (Moving Picture Expert Group), která patří pod organizaci ISO/IEC a skupiny VCEG (Video Coding Experts Group), která spadá pod organizaci ITU-T.

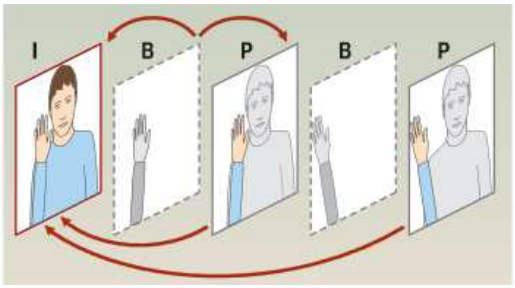
Princip standardu H.264 je velmi podobný fungování standardu MPEG-2. Video je rozděleno na jednotlivé snímky, které jsou následně zpracovávány. Rychlost dělení videa na snímky se pohybuje kolem 24 až 30 snímků za sekundu. Snímky jsou dále rozděleny do GOP (Group Of Picture) skupin. Průměrný počet snímků ve skupině se pohybuje kolem 12. První snímek v GOP skupině se nazývá I-snímek. Tento snímek je zkomprimován jako běžný obrázek a rozdělen na makrobloky 16x16 pixelů. Každý makroblok je poté převeden do prostoru YCrBr, který se dělí na jas, modrou složku a červenou složku, jak je viditelné na obrázku 1.1. Jasová složka má větší prostor pro informaci, protože lidské oko je více citlivé na změnu jasu než na změnu barvy. [4]



Obrázek 1.1: *Jasová a barvonosné složky*

Makrobloky se charakterizují co nejúsporněji, takže pro celý makroblok může existovat pouze jedna hodnota barvy a jedna hodnota jasu. Kvalitu obrazu ovlivňují dva faktory – obsah komprimovaných pixelů a velikost datového toku. Pokud je komprimovaný obsah velmi podobný (typicky jednobarevná plocha), není zjednodušení rozpoznatelné a obraz vypadá přirozeně. Problém nastává u složitějších obsahů. Kvalitu složitějších makrobloků ovlivňuje dostupná velikost. Máme-li dostatečné množství prostoru, můžeme se věnovat jednotlivým pixelům složitého obrazu a kvalita je vyšší. Při malém množství bajtů se jednotlivé detaily ztrácí.

Komprimování ostatních snímků je možno provést dvěma metodami a to pomocí P snímků nebo B snímků. Metoda P snímků (Predicted) je založena na předchozím snímku, který je považován za snímek referenční. Při této metodě zaznamenáváme pouze změny v makroblocích oproti předchozímu snímku. B snímky (Bidirection Predicted) jsou založeny na podobném principu jako P snímky. Určují rozdíl oproti předchozímu snímku, ale umí využít i rozdílů oproti následujícímu snímku. Výhoda této metody je, že si můžeme vybrat snímek, se kterým bude obraz kvalitnější a díky tomu bude třeba méně prostoru pro uložení této informace. Rozdíly mezi jednotlivými typy snímků jsou znázorněny na obrázku 1.2. [5]



Obrázek 1.2: *Rozdíl mezi B a P snímky*

H.264 používá celočíselnou transformaci na blocích o velikosti 4 × 4, případně o velikosti 8 × 8. Menší bloky se vyplatí především tehdy, když jsou na snímku velké rozdíly a značné detaily. Navíc i ostré kontrastní předměty v obraze jsou velmi dobře vykresleny.

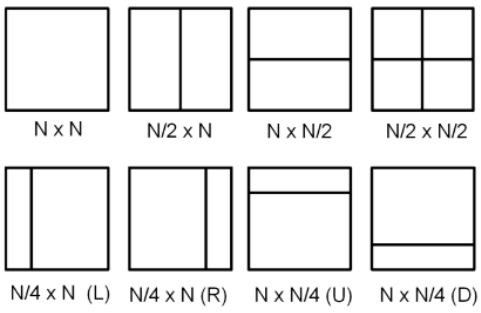
Při kompresi jsou všechny hodnoty zaznamenávány jen jako rozdíl oproti sousednímu bloku. Jelikož kodér začíná kódovat snímek vždy vlevo nahoře, je toto místo bráno jako referenční pro celý makroblok. Pokud se jedná o rovnoměrnou plochu, jsou hodnoty jasu i barvy identické. V tomto případě může kodér frekvence jednoho bloku odvodit od levého nebo horního souseda. Po kvantizaci jsou hodnoty plné nul, což je optimální případ pro komprimaci. Dokonce, i když nejsou hodnoty jasu a barev zcela identické, jsou si hodnoty diferencí podobné, což je mnohem příznivější pro kompresi, než když se musí komprimovat celé hodnoty. Pro prostorovou predikci používá H.264 proměnnou velikost bloku 4 × 4, 8 × 8 nebo 16 × 16.

1.2.2 **H.265**

Od roku 2013 je k dispozici nástupce standardu H.264, označovaný jako H.265 nebo také HEVC (High Efficiency Video Coding). Ten slibuje dvakrát větší účinnost komprimace, což znamená, že při srovnání s H.264 budeme například potřebovat pouze poloviční šířku pásma při stejné kvalitě obrazu, případně při stejném datovém toku může být video dvakrát tak kvalitní.

H.265 predikce se oproti předešlému H.264 standardu liší pouze nepatrně. Využíváme i velké části sousedních snímků pro intra-frame predikci, která se využívá k určení rozdílů hodnot, tímto zvyšujeme účinnost kódování. Nadále je používána také inter-frame predikce, která využívá podobnosti uvnitř makrobloku. S přibývajícími možnostmi se nám však zvyšují nároky na výpočetní výkon, proto se snímky zpracovávají paralelně. Paralelně zpracováváme několik snímků ve stejném časovém okamžiku.

H.265 je vytvořen aby hledal co nejvíce snímků se stejnými nebo podobnými parametry, protože hodnoty stačí uložit pouze jednou a hodnoty velmi podobných snímků jednoduše odvodíme. Pro fungování tohoto systému je však třeba vysoký výpočetní výkon, protože je třeba flexibilně měnit velikost a tvar bloků. Velikost bloků je ovlivněna jejich obsahem, například v místě s podrobnějšími detaily jsou používány menší bloky a naopak, kde se ve snímku nachází část s jednotnou barvou, můžeme použít blok větší. Na výběr máme 8 typů PB (Prediction Blocks)viditelné na obrázku 1.3, které se liší odlišným rozdělením na jednotlivé menší bloky. Typ vybíráme podle jednotlivých částí bloku. [6]



Obrázek 1.3: *Jednotlivé typy PB*

1.2.3 **VP8**

Jedná se open-source projekt, který v roce 2010 převzala firma Google od výrobce On2 Technologie, který VP8 představil poprvé v roce 2008. Firma Google financovala rozvoj tohoto standardu a podporu pro jednotlivé webové prohlížeče s podporou HTML5. Standard byl primárně vytvořen jako alternativa standardu H.264. Jeho nespornou výhodou je BSD licence, která povoluje volné šíření obsahu, proto je tento standard často využíván v open-source projektech.

Vývoj byl již z počátku situován do oblasti webových aplikací, a proto standard předpokládá omezenou šířku pásma a rozmanitost cílového zařízení. Podobně jako u standardu MPEG využíváme převodu do prostoru YCbCr o jasové složce s velikostí makrobloku 16x16 pixelů a chrominanční složky mají makroblok o velikosti 8x8 pixelů.

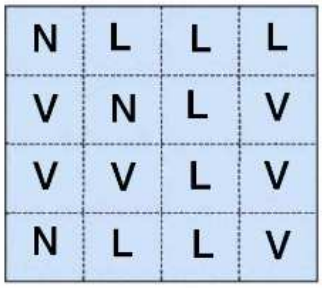
Rozdílné jsou oproti MPEG standardům typy snímků. Standard VP8 nevyužívá snímky typu B a mohou být použity pouze tři referenční snímky. U VP8 se I snímek označuje jako key frames. Pro interpredikci využíváme tři druhy referenčních snímků. První variantou je golden reference frame, který se ukládá do bufferu. Využívá se především pro statické pozadí scény, na které v popředí probíhá pohyb. Další využití je při změnách z jedné scény na druhou a zpět.

Druhým typem je last frame, který využívá předchozích snímků. Třetím typem je alternate reference frame, který se používá převážně jako referenční snímek pro zvýšení přesnosti ostatních snímků. Je vytvářen z několika golden frame snímky za účelem co největší komprese.

VP8 používá diskrétní kosinovu transformaci s bloky o velikosti 4x4 pro jasové i chrominanční složky. Pro kvantizaci máme pro stejnosměrné i střídavé složky chrominančních i jasových koeficientů různé kvantizační hladiny. Celkem je definováno 128 hladin a tento typ kvantizace se nazývá adaptivní.

VP8 využívá několik druhů predikce. Jednou z těchto predikcí je horizontální predikce, která využívá makrobloků nalevo od kódovaného makrobloku. Další možností je predikce DC koeficientů. Tato predikce využívá průměr hodnot pixelů nad a nalevo od kódovaného makrobloku. Jednou z nejčastěji používaných metod predikce je metoda TrueMotion, která používá predikci pohybu na základě řádku nad makroblokem, sloupce vlevo a pixelu vlevo nad makroblokem

V VP8 jsou pro každý inter predikovaný snímek k dispozici až tři dříve zakódované snímky. Navíc byl vyvinut flexibilní mód pro interpredikci nazvaný SPLITMV. Ten umožňuje rozdělení jasových makrobloků 16x16 do 16 subbloků o velikosti 4x4 a je vhodný v případě, že makroblok má různé pohybové vlastnosti. Každému subbloku pak připadá jeden vektor pohybu. Subbloky však mohou tento vektor převzít od sousedního subbloku výše nebo nalevo, což způsobuje další zvýšení komprese. Na obrázku 1.4 je viditelný příklad, ve kterém písmeno N označuje nový pohybový vektor, Písmeno L znamená vektor od bloku nalevo a písmeno V od bloku výše.

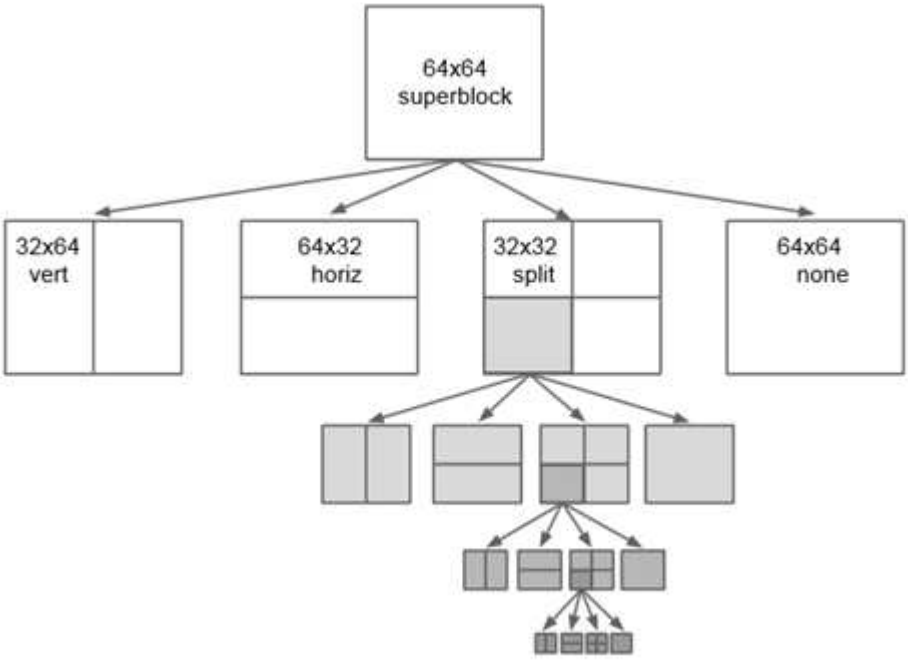


Obrázek 1.4: *Možnost predikce vektoru pohybu*

Rekonstrukční filtr je adaptivní. Jeho typ a vlastnosti jsou upravovány pro různé predikční módy a referenční snímky. Například je umožněno použít silnější filtr na některé části bloku a zároveň použít i měkký filtr na zbylé části bloku. [7][8][9]

1.2.4 **VP9**

Vývoj standardu VP9 začal ve třetím čtvrtletí roku 2011 a hlavním cílem bylo snížit přenosovou rychlost oproti VP8 o 50 procent, při stejné kvalitě videa. VP9 využívá superbloky o velikosti 64x64 pixelů, které je možno rozdělit na jednotlivé menší bloky až o velikosti 4x4 pixelů. Superbloky lze rozdělovat několika způsoby. Bloky můžeme dělit vertikálně, horizontálně, nebo pomocí split dělení, které děli i horizontálně i vertikálně. Jednotlivé možnosti dělení jsou znázorněny na obrázku 1.5.



Obrázek 1.5: *Možnosti dělení superbloků*

Pro inter-mode, což je predikce oproti sousedním snímkům se užívá DCT (discrete cosine transform) transformace. Pro intra-mode se užívá hybridní DCT/ADST (discrete cosine transform) transformace.

Nová vlastnost VP9 je velikost referenčních rámců. Každý nový snímek může být zakódován jiným rozlišením jak ten předešlý.

Segmentace je založena na sdružování bloků, které mají stejné charakteristiky. Bloky jsou určovány pomocí ID.[10]

1.3 Vývojové prostředí pro multimédia ve webovém prostředí

Jedná se především o zásuvné moduly, které jsou určeny k podpoře multimédií. Jejich úkolem je podporovat přehrávání videa, počítačových her a interaktivniho multimediálního obsahu ve webovém prohlížeči. Jedná se především o dva hlavní představitele a to Silverlight a Flash Player. Loga jednotlivých projektů jsou znázorněny na obrázku 1.6.

Obrázek 1.6: *Loga jednotlivých platforem*

1.3.1 Microsoft Silverlight

Jedná se o platformu pro aplikace, která byla vytvořena firmou Microsoft. Je určena především pro vývoj multimediálních aplikací a umožňuje těmto aplikacím fungovat v rámci webového prohlížeče, nebo svém vlastním okně. Silverlight umožňuje tvořit Rich Internet Applications (RIA) pomocí standardních nástrojů jako je XAML a C#, nebo Javascriptu. Silverlight umožňuje vytvářet interaktivní obsah, který běží u klienta, a podporuje dynamické grafiky, média a animace, které daleko přesahují možnosti obyčejného HTML. Platforma se do prohlížeče zavádí jako zásuvný modul a podporuje rozličné množství webových prohlížečů i operačních systémů.

První verze této platformy se nazývala Windows Presentation Foundation/Everywhere a byla představena v roce 2007. Platforma obsahovala vlastní uživatelské rozhraní, grafiku, animace, atd. a skládala se z jednotlivých součástí:

- Input (vstup) - ovládání vstupu zařízení
- UI core (jádro uživ. rozhraní) - spravování a vykreslování obrázků
- Media - přehrávání audia a Videa
- XAML - programovací jazyk k úpravě/vytvoření nového uživ. rozhraní.

Druhá verze se objevila později v roce 2007 a byla doplněna o velmi důležité prvky při tvorbě aplikací. Jednalo se především o zaškrtávací pole, tlačítka, seznam, přepínač, indikátor průběhu a další prvky pro vytvoření lepšího uživatelského rozhraní. Dalším důležitým prvkem je funkce DeepZoom, technologie umožňující přirozeně přiblížit a oddálit obrázek kolečkem myši.

U verze 3 se objevuje podpora standardu pro video H.264 pro přehrávání HD videa. Umožňuje propojení jednotlivých ovládacích prvků a jejich vzájemnou interoperabiltu a dále také přesměrování pomocí hypertextových odkazů a tzv. deep-linking, které je určeno pro odkazování přímo na specifickou stránku.

Verze 4 nově nabízí využití webových kamer a mikrofonů, plnou podporu tisku, 60 nových ovládacích prvků, vylepšení technologie DeepZoom, podpora off-line přehrávání obsahu. Byly vylepšeny také aplikace fungující mimo webový prohlížeč. Je zavedena plná podpora klávesnice při režimu na celou obrazovku a je umožněn přístup k uživatelským dokumentům.

Poslední verze 5 byla představena v roce 2011. Jedná se o aktuálně dostupnou verzi programu. Mezi inovace v 5 verzi patří například rychlejší start aplikace, zlepšená podpora dálkových ovladačů, využití procesoru grafické karty, zjasnění textu za účelem lepší viditelnosti a především podpora 64 bitových systémů.

Jako vývojové prostředí se především používá Visual Studio, které je také produktem firmy Microsoft, kde nevýhodou je pořizovací cena. Dále je také možné použít i open-source řešení např. Eclipse4SL.

Oproti Flashi umožňuje vyvíjet program pomocí programovacího jazyka C#. Což dovoluje vývojářům používat stejný programovací jazyk jak na straně klienta, tak i na straně serveru. Silverlight je založen na časové ose, kdežto Flash na snímcích, což usnadňuje práci s animacemi. Dále také podporuje fullscreen video s podporou změny velikosti obrazu. Silverlight umožňuje vložení fontů přímo do projektu

Podporované funkce:

- Velmi široká podpora prohlížečů
- odlehčený zásuvný modul – snadná a rychlá instalace je výhodou pro uživatele
- Dvourozměrné kreslení – lze dokonce spouštět akce kliknutím na část grafiky a tím dodávat webovým stránkám větší interaktivou
- CLR - V mnoha případech mohou vývojáři vzít kód původně napsaný pro úplný CLR .NET a po mírných změnách ho použít v aplikaci Silverlightu.
- Interakce s webovými službami

Silverlight podporuje standardní kodeky na přehrávání populárních formátů, jako MP3, WMA, WMV, MPEG, VC-1. [11][12]

1.3.2 Adobe Flash player

Je multimediální platforma, která se začala vyvíjet původně pro tvorbu křivkových animací. Prvotní formou flash byl kreslící program SmartSketch původně založený na programovacím jazyku Java. Původní projekt koupila firma Macromedia a začala platformu rozvíjet do dnešní podoby. Prvotně se platforma jmenovala Macromedia Flash a až v pozdější době se objevuje název Adobe Flash. V dnešní době se jedná o platformu pro tvorbu multimediálních aplikací jako jsou animace, hry a především k tvorbě RIA (Rich Internet Aplication) aplikací. Stejně jako předešlá platforma je Adobe Flash Player vytvořen jako malý zásuvný modul do webového prohlížeče, nebo jej můžeme používat zvlášť ve vlastním okně, této variantě se říká Adobe Air. Adobe Flash především umožňuje:

- podporovat 64-bitové systémy
- přehrávání audia, videa a animací
- používat aplikace i mimo webový prohlížeč
- vytvářet grafické aplikace za užití vektorové grafiky
- interaktivita a tím celkové oživení webových stránek

Adobe Flash využívá zcela jiný programovací jazyk pojmenovaný ActionScript a jako programovací prostředí využíva software Flex. ActionScript je objektově orientovaný programovací jazyk vytvořený pro realizaci programové logiky u aplikací Adobe Flash. Tento programovací jazyk vychází z koncepce programovacího jazyku JavaScript.

Nejčastějším formátem Flash souborů je SWF formát, který známe pod názvem ShockWave Flash. Tyto souboru mají obvykle příponu .swf a jsou používány ve formě pluginu do webového prohlížeče a následně se spouští pomocí Adobe Flash Playeru.

Adobe Flash podporuje video kodeky standardu H.264, které nabízí vysokou kvalitu videa při nízkých přenosových rychlostech. Dále také kodeky standardu MPEG-4 a také On VP6 kodek. [13]

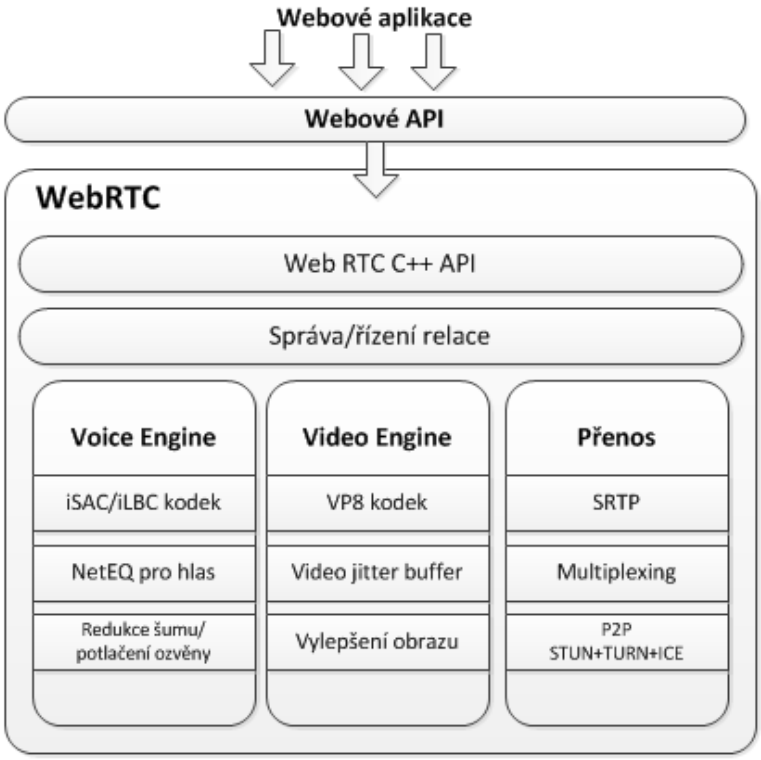
2 WebRTC technologie a SIP B2BUA

Technologie WebRTC může být realizována i pomocí pobočkových ústředen Asterisk, které jsou založeny na SIP směrovacím protokolu. Proto je nutné nejprve pochopit princip fungování samotné Asterisk ústředny, abychom byli schopni implementovat WebRTC technologii.

2.1 WebRTC

Web pro Real-Time Comunication se využívá pro komunikaci mezi internetovými prohlížeči v reálném čase. Jedná se o open source projekt, který je otevřený a volně šiřitelný. Komunikace může probíhat mezi různými typy internetových prohlížečů, ať už se jedná o Firefox, Google Chrome nebo Operu. Jedinou podmínkou pro úspěšnou komunikaci je podpora HTML verze 5. Tato vyšší verze HTML je nutná díky podpoře multimediálního obsahu, v našem případě přehrání audia a videa přímo v internetovém prohlížeči bez potřeby Flash a jiných zásuvných modulů. Nyní WebRTC podporuje videokodeky typu VP8 od firmy Google a jako druhá varianta je videokodek typu H.264. Jako audiostandardy jsou povoleny iLBC a iSAC. Některé zdroje uvádí ještě audiokodeky G.711 a G.722. [15]

Pro realizaci této komunikační platformy je potřeba Java Appletu, který je implementován přímo v prohlížeči. Jedná se aplikaci spustitelnou přímo z internetového prohlížeče a poskytuje nám vzájemnou funkci našich internetových prohlížečů s HTML verze 5. Další výhodou WebRTC je necitlivost na typ operačního systému, jelikož je implementován v prohlížeči. Spojení mezi uživateli je peer-to-peer pro propojení dvou účastníků. Technologie WebRTC přináší zcela nový tip architektury, která je znázorněna na obrázku 2.1.



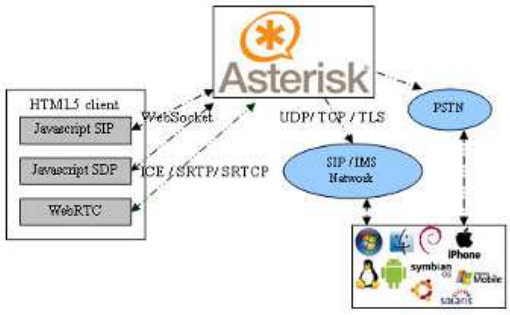
Obrázek 2.1: *Architektura WebRTC technologie*

- **Webové aplikace** - Různé webové aplikace umístěné v prohlížeči, které jsou využívány pro WebRTC technologii, tudíž pro multimediální komunikaci.
- **Webové API** - Obsahuje sadu příkazů psaných v jazyce JavaScript. Tyto příkazy jsou určeny pro využití možností technologie WebRTC.
- **WebRTC C++ API** - Jedná se o část API určenou pro spojení peer-to-peer. Toto spojení se využívá pro spojení mezi klienty a výměnu multimediálních dat.
- **Voice engine** - Technologie určená pro získání audio signálu. Je využita zvuková karta a její data jsou poté přenášena do sítě. Technologie obsahuje potřebné kodeky pro kódování audia i kodeky pro hovor. Technologie také obsahuje nástroj pro minimalizaci okolního šumu a ozvěn.
    - **iSAC** - je širokopásmový a super širokopásmový kodek pro IP komunikaci VoIP a audio. Kodek iSAC využívá vzorkovací frekvenci 16 kHz nebo 32 kHz. Je možnost také variabilní přenosové rychlosti od 12 až do 52 kbps.
    - **iLBC** – oproti předešlému kodeku se jedná o úzkopásmový hlasový kodek pro VoIP a audio. Vzorkovací frekvence je 8 kHz. A rychlost je závislá na velikosti rámce: 15,2 kbps pro 20 ms rámce a 13,33 kbps pro 30 ms rámce.
    - **Opus** – je kodek s variabilní velikostí rámce (2,5 ms až 60 ms) i rychlostí od 6 kbit/s do 510 kbit/s.
    - **NetEQ pro hlas** - Dynamický jitter buffer a algoritmus pro odstranění chyb určený pro potlačení nepříznivých účinků síťového rušení (jitter) a také ztrátovosti paketů.
    - **Acoustic Echo Canceler (AEC)** - je komponent který odstraňuje akustické echo způsobené zvukovou vlnou od aktivního mikrofonu. Odstranění echa je prováděno v reálném čase.
    - **Noise Reduction (NR)** - jedná se o software vytvořený pro odstranění šumu vzniklého většinou okolním prostředím.
- **Video Engine** - Technologie určená pro získání obrazu z kamery nebo fotoaparátu a následný přenos do sítě nebo zobrazení na obrazovce. Kromě kodeků je zde také nástroj pro vylepšení obrazu, který odstraňuje obrazový šum. Video jitter buffer pomáhá potlačit chvění a ztrátu paketů.
    - **VP8**- video kodek určený pro kódování videa v reálném čase.
    - **Video Jitter Buffer** – jde o dynamický buffer a je určen pro potlačení účinků chvění a ztrátovosti paketů. Pomáhá k vylepšení celkové kvality obrazu.
    - **Vylepšení obrazu** – je docíleno odstraněním obrazového šumu ze záběru webkamery.
- **Přenos** - Pro komunikaci je využíván zabezpečený real-time protokol SRTP. Je zde využito také možnosti multiplexování, což je sloučení více komunikací do jednoho datového toku. V neposlední řadě obsahuje nástroje pro spojení různých typů sítí a překonání NAT. Jako jsou STUN, což je kolekce standardů, které umožňují komunikovat přes NAT. Podobnou funkci má i ICE.

Pro komunikaci pomocí technologie WebRTC může uživatel využít již vytvořených webových rozhraní např, SipML5 klienta. Nebo může vytvořit vlastního klienta založeného na HTML5 a JavaScriptu, jak je popsáno v [18].

2.2 SipML5

Je první HTML5 SIP klient, vytvořený jako open-source projekt pomocí programovacího jazyka JavaScript. Využívá webrtc2sip bránu, která spojuje funkce WebRTC a SIPu pro vytvoření telefonního zařízení v rámci webového prohlížeče. Umožňuje webovému prohlížeči přijímat hovory a vysílat hovory do SIP sítě jak je znázorněno na obrázku 2.2. Výhodou je, že není třeba žádného zásuvného modulu, je jenom třeba podpory WebRTC. [16][17]



Obrázek 2.2: *Architektura SIPml5*

SIPML5 podporované prvky:

- Je podporován webovými prohlížeči Chrome, Firefox, Internet Explorer, Safari a Opera
- Audio a Video hovory, podpora více hovorů a více účtů
- Sdílení plochy z Chrome prohlížeče jakémukoliv SIP klientovi
- Instant messaging – posílání zpráv mezi prohlížeči
- Přidržení hovoru a obnovení, předání hovoru
- Tónová volba (DTMF) pomocí SIP INFO
- Funkce Click-to-call
- SIP TelePresence – Skupinový videochat

2.3 SIP

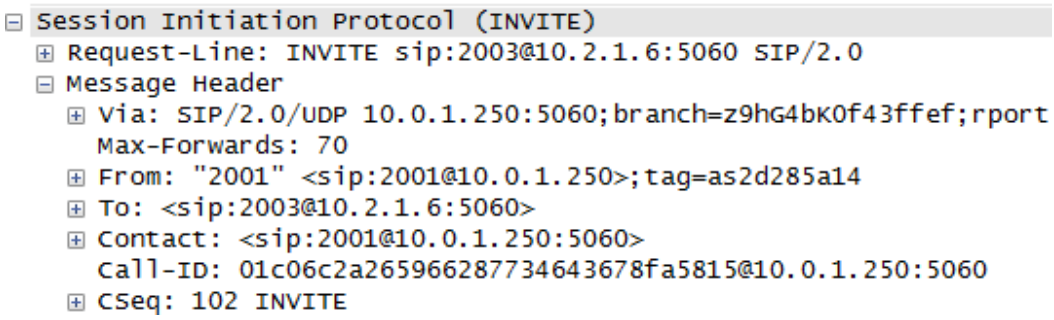
Jedná se o řídící protokol pracující na aplikační vrstvě. Je určen pro sestavení, modifikaci a ukončení spojení mezi účastníky. Jedná se o typ spojení end-to-end, což znamená, že veškerá logika je uložena v koncových zařízeních. Dále je tento protokol textově orientovaný, což je velkou výhodou pro snadnější orientaci ve spojení. Pro směrování je využíváno hlaviček paketů, kde hlavička paketu sestavující spojení je na obrázku 2.3.

Protokol je celkově realizován pomocí struktury klient-server, takže pro komunikaci využíváme dva typy zpráv. Klient posílá směrem k serveru žádosti a server reaguje ve formě odpovědí. V našem případě je klient určitý SIP telefon, ať už hardwarový nebo softwarový. A jako server jsou SIP servery, které poskytují klientovi další služby sítě jako je registrace, účtování, lokalizace a mnoho dalších.

Základní typy žádostí:

- **REGISTER** – Je žádost o registraci vůči serveru a ověření klienta
- **INVITE** – jedná se o žádost o sestavení spojení, v těle této zprávy jsou uloženy parametry spojení
- **ACK** – se používá pro potvrzení předem přijaté odpovědi
- **BYE** – tuto zprávu vysílá strana, která se rozhodne ukončit spojení
- **CANCEL** – je pro ukončení ještě nesestaveného spojení (ještě nebyla vyslána odpověď 200 OK)
- **OPTIONS** – zjišťuje jaké má protistrana možnosti nastavení (kodeky, typy médií, …)

Typická SIP žádost vypadá následovně:



Obrázek 2.3: *SIP hlavička žádosti*

- **Request-line** – určuje typ zprávy (INVITE) a request URI, které obsahuje další krok zprávy
- **Via** – zobrazuje použitý transportní protokol, adresu a port proxy serveru a hodnotu branch, který se používá jako identifikátor transakce
- **Max-Forwards** – maximální počet kroků v síti, každý prvek v sítí číslo sníží o jedna, slouží proti nekonečným smyčkám
- **From** – zobrazuje adresu volajícího, klienta, který posílá žádosti
- **To** – adresa volaného, přijímá žádosti a zasílá odpovědi
- **Contact** – zobrazuje kontakt na UA
- **Call-ID** – unikátní identifikátor dialogu, každý dialog má jinou hodnotu parametru
- **Cseq** – Určuje pořadí v dialogu a každá nová transkace kromě ACK a CANCEL navyšuje hodnotu

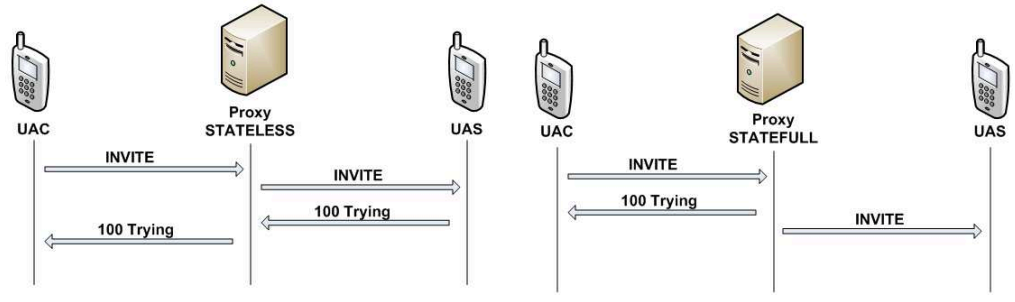
Jestliže obdržíme žádost, je třeba odeslat odpověď. Odpověď se neposílá pouze na metodu ACK, která nám potvrzuje žádost INVITE. Můžeme však použít metodu PRACK (Provisional Acknowledgement), která se na rozdíl od ACK potvrzuje.

Základní typy odpovědí jsou rozdělovány do kategorií podle svého kódu:

- **1xx** – Dočasné informativní odpovědi, které jsou odesílány na přijaté žádosti není-li ještě znám výsledek zpracování. Mezi nejpoužívanější patří 100 Trying a 180 Ringing, které informují o sestavování spojení
- **2xx** – Jsou konečné pozitivní odpovědi. Poslední odpověď na žádost o spojení, které oznamují úspěšné vyřízení žádosti. Nejznámější odpovědí je 200 OK, které signalizuje navázání spojení.
- **3xx** – Odpovědi určené pro přesměrování. Udávají parametry přesměrování a informace o novém cílovém zařízení. Tyto odpovědi jsou konečné.
- **4xx** – Signalizují chybu na straně klienta a jedná se o negativní konečné zprávy. Většinou se jedná o chybnou syntaxi.
- **5xx** – Chyba na straně serveru. Není problém v syntaxi, ale server zřejmě nedokázal požadavek zpracovat.
- **6xx** – Poslední z možných chybových hlášení je globální chyba. Je generována není-li možno požadavek zpracovat žádným dostupným serverem.

Síť SIP má své základní prvky:

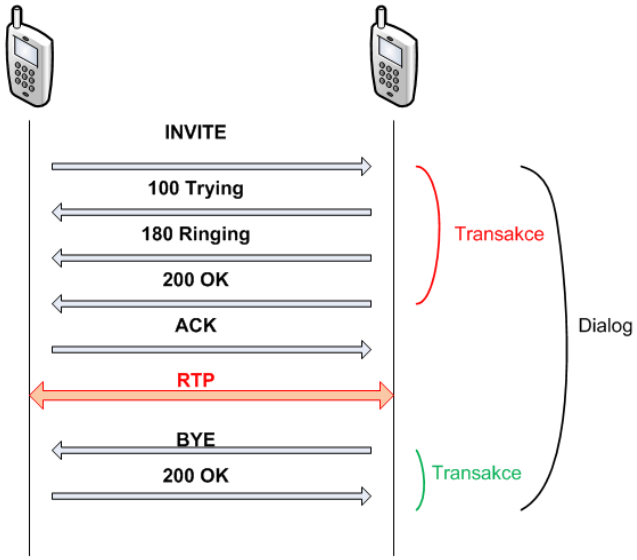
- **UA (User Agent)** – jedná se o koncové terminály, hardwarové nebo softwarové SIP telefony
  - **UAC (User Agent Client)** – jde o vysílací stranu, která vysílá žádosti směrem k UAS
  - **UAS (User Agent Server)** – přijímající strana, zpracovává žádosti a odesílá odpovědi zpět UAC
  - **B2BUA (Back-to-back User Agent)** – jedná se o speciální typ klienta, který vytváří nové spojení
- **SIP server**
  - **Proxy** – hlavní směrovací server, který se podle metody posílání zpráv dělí na dva typy, které jsou znázorněny na obrázku 2.4:
    - **Stateless** – Bezestavový server, který nemá v paměti předešlé stavy, zprávy pouze přeposílá. Je rychlejší než Stateful SIP Proxy . Využívá se především pro jednoduché směrování a překládání zpráv.
    - **Stateful** – Má informace o předešlých stavech, zprávy posílá dále až po předběžném vyřízení. Po přijetí požadavku, server vytvoří záznam stavu a drží důležité informace, dokud nedojde k ukončení transakce (dokud není vyřízena žádost) či dialogu (dokud není ukončeno celé spojení
  - **Redirect** – server určený pro přesměrování, zde získáme nové informace o uživateli
  - **Registrar** – je server s registrační databází a registruje uživatele vůči serveru
  - **Location** – server s informacemi o umístění uživatelů a serverů



Obrázek 2.4: *Rozdíl mezi stateless a statefull proxy serverem*

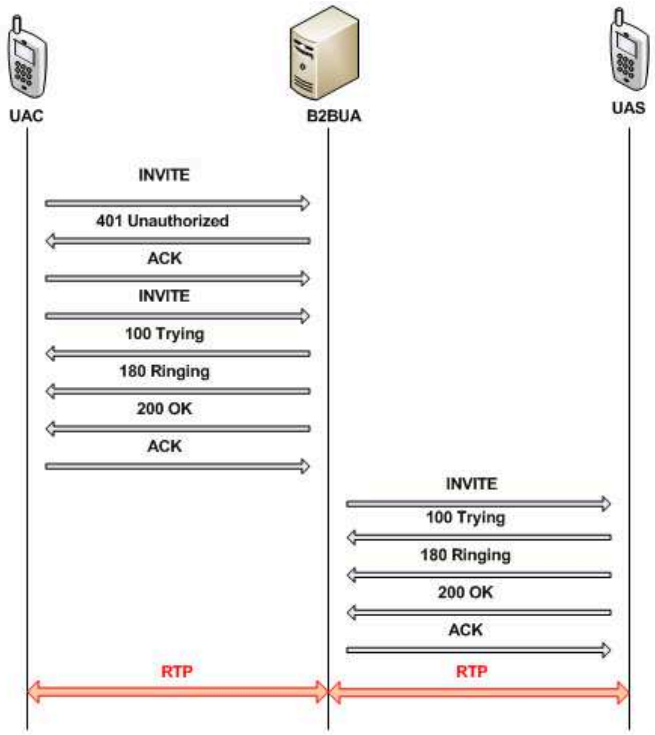
SIP komunikace se dělí na Transakce a Dialogy. Transakce je skupina SIP zpráv, které jsou vyměňovány mezi SIP síťovými prvky. Transakce obsahuje jednu žádost a všechny odpovědi na ni navazující. Identifikátor umožňující porovnání transakcí se nazývá branch a nachází se v poli Via SIP hlavičky, viz obrázek 2.5.

Dialog jsou SIP zprávy probíhající mezi dvěma komunikujícími body. Dialogy jsou identifikovány pomocí polí ze SIP hlavičky nazvané From(tag), To(tag) a Call-ID. Mají-li hlavičky stejné hodnoty, tak tyto zprávy patří do jednoho dialogu.



Obrázek 2.5: *Ukázka transakce a dialogu*

Již dříve zmíněný B2BUA je speciální typ User Agenta, který nejprve vytvoří spojení s jednou stranou a až poté navazuje spojení směrem k druhému uživateli, jak je viditelné na obrázku 2.6. Výhodou je že koncový uživatel nepozná rozdíl mezi rozdíly spojení pomocí B2BUA nebo proxy. B2BUA má zcela rozlišné chování oproti SIP proxy serveru, který zprávy přeposílá, na rozdíl od B2BUA, které vytváří nové zprávy. B2BUA má větší možnosti než proxy, avšak co se týče rychlosti obsloužení je pomalejší. Mezi nejznámější představitele B2BUA patří software Asterisk, proxy server zastupuje většinou software Kamailio. [14][15]



Obrázek 2.6: *Ukázka směrování u B2BUA*

2.4 Průchod SIP komunikace přes NAT

NAT (Network Acess Translator) pracuje s IP adresami a porty sítě a na základě definovaných parametrů překládá veřejné IP adresy na adresy privátní. Této funkce se využívá většinou pro zvýšení bezpečnosti ve firemních a soukromých sítích. Problém poté nastává při sestavování hovoru pomocí SIP protokolu, který pracuje s předem definovanými IP adresami a porty. Kvůli těmto komplikacím existují mechanismy, které podporují IP telefonii i v síti s NAT prvkem.

Jednou z možností je využití STUN (Session Traversal Utilities for NAT). Tento mechanismus má však potíže s překonáním symetrického NAT. Princip je založen na dotazech klienta směrem k STUN serveru, který je umístěn ve veřejném IP prostoru. Klient poté obdrží odpověď ohledně IP adresy a použije jí do hlavičky SIP zpráv. Velkou výhodou je podpora i vícenásobných NAT brán. Problém nastává pouze v případě symetrického NAT mapování, kdy se liší porty pro komunikaci se STUN serverem a SIP serverem, který potřebujeme pro směrování zpráv.

Rozdílné typy NAT podle STUN:

- **Full-cone NAT** – Všechny požadavky z jedné vnitřní IP adresy a portu jsou vždy mapovány na stejnou externí IP adresu a port.
- **Address-restricted-cone NAT** – Na rozdíl od Full-cone NAT může host z externí sítě zaslat paket lokálnímu počítači, pouze pokud lokální počítač již dříve komunikoval s tímto externím hostem.
- **Port-restricted cone NAT** – Funguje podobně jako Address-restricted-cone NAT, ale je omezen čísly portů. Pakety lze z vnější sítě posílat jen přes ty porty, které byly již předem použity pro komunikaci směrem z vnitřní sítě.

- **Symmetric NAT** – Veškeré požadavky ze stejné lokální IP adresy a portu na určitou IP adresu a port jsou mapovány na jedinečnou externí zdrojovou IP adresu a port. V případě paketu ze stejného lokálního zařízení a stejné zdrojové adresy a portu, ale rozdílného cílového portu a adresy je použito jiné mapování. Posílat pakety z externí sítě můžou jen ty zařízení, které již předem obdržely paket z vnitřní sítě.

TURN je rozšíření STUN metody, a užívá se jsou li oba koncové body skryté za NATem. TURN využívá protokolu STUN, ale definuje nové žádosti, parametry a celkově nový funkční možnosti. TURN je stavový protokol, který definuje relace a každá relace má svoji vlastní dobu trvání a kontext.

Vzhledem ke složitému překládání IP adres, nárokům na šířku pásma a na přidaném zpoždění při přenosu médií se TURN využívá jako poslední možnost.

ICE (Interactive Connectivity Establishment) je protokol, který kombinuje metody STUN a TURN. Umožňuje obsluhovat pakety na základě priorit, které vypočítá na základě analýzy sítě a možností.

Podstatnou výhodu při pužívání NAT má B2BUA, který má plnou kontrolu nad spojením včetně médií. Podpora směrování přes NAT je velmi jednoduše implementovatelná. Stačí pouze úpravy v konfiguračním souboru /etc.asterisk/sip.conf , ve kterém nastavíme příkazy v tabulce 2.1.

Tabulka 2.1: *Příkazy pro podporu NAT v sip.conf*

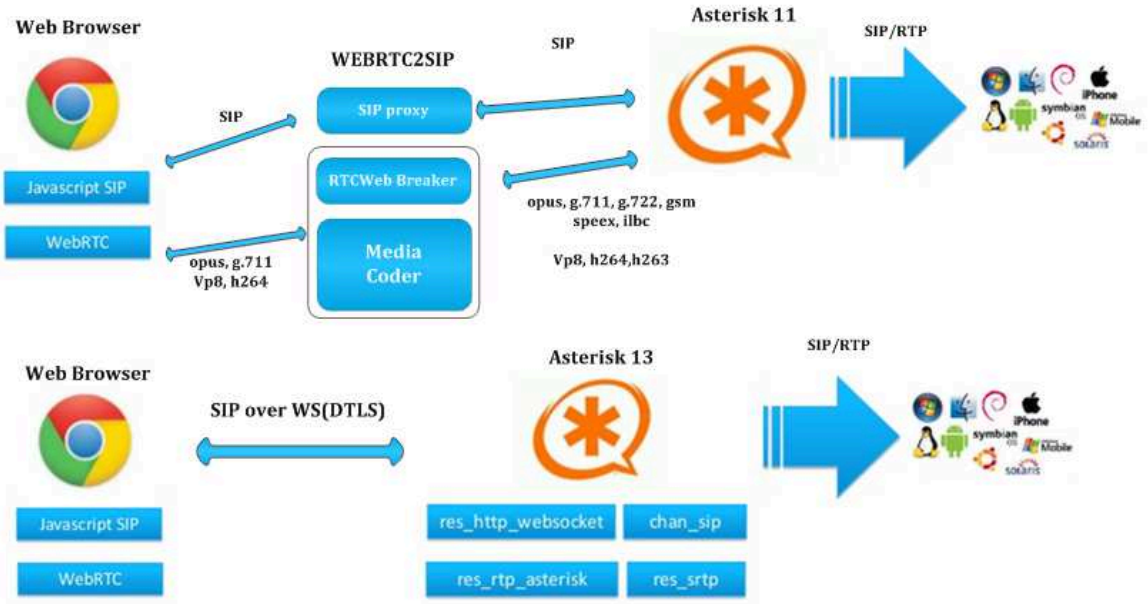
příkaz	**význam**
nat=yes	Asterisk nevyužívá IP adresy a portu které zasílá klient, ale odpovědi posílá zpět na IP adresu a port ze které obdržel žádost
qualify=yes	Je určeno pro pravidelné zasílání zpráv SIP OPTION pro získání mapování adres
directmedia = no	Je třeba zakázat možnost re-invite, možnost opakovaného pokusu o sestavení spojení. U STUN se nezakazuje.

3 Praktická realizace různých scénářů nasazení WebRTC

V následující kapitole porovnávám prvotní řešení WebRTC, které se objevilo v roce 2012 a bylo řešeno především pomocí Asterisk 11 s novější variantou řešení užití Asterisk 13. Hlavní rozdíl je ve využití webrtc2sip brány, kde Asterisk 11 nemá potřebné mechanismy pro podporu potřebných audio a video kodeků, proto využívá tuto bránu pro překlad na jiné podporované kodeky. Oproti tomu u Asterisk 13 jsou již tyto problémy vyřešeny, tím už není nutné využití této brány pro překlad. To se projeví i počtem instalací balíčků, které jsou třeba pro fungování WebRTC. V prvotní verzi bylo potřeba doinstalovat velké množství balíčků, především pro překlady a podpory kodeků. V následující tabulce 3.1 je znázorněn postup jednotlivých instalací. Hlavní rozdíly mezi oběma variantami jsou především ve využití překladu kodeků, jak je znázorněno na obrázku 3.1.

Tabulka 3.1: *Postup jednotlivých instalací*

WebRTC s Asterisk 11	WebRTC s Asterisk 13
YASM	jasson
OPUS	
ffmpeg	Asterisk 13
doubango	
webrtc2sip	Generování klíčů a certifikátů
Asterisk 11	
Generování klíčů a certifikátů	Nastavení PBX asterisk
Nastavení PBX asterisk	



Obrázek 3.1: *Rozdíly mezi jednotlivými scénáři*

3.1 WebRTC s Asterisk 11

Instalaci jsem prováděla na vzdáleném virtuálním počítači, který byl vytvořen pro účely mé diplomové práce. Pro vzdálený přístup jsem využila protokolu ssh. Po připojení na vzdálený počítač bylo nejprve nutné doinstalovat jednotlivé knihovny, které jsou potřeba k fungování následně doinstalovaných balíčků. V první fázi se jedná o tyto balíčky:

Tabulka 3.2: *Potřebné balíčky pro instalaci*

Balíček	**Podpora**
`libssl-dev`	pro následnou instalaci webrtc2sip brány
`libsrtp0-dev`	
`libxml2-dev`	
`libtool`	
`libspeexdsp`	pro audio kodeky
`libvpx`	pro VP8
`libx264`	pro H.264

3.1.1 YASM a OPUS

Jako první je třeba nainstalovat YASM balíček, který je nutný, jestliže plánujeme využívat VP8 a H.264 kodeky. Balíček je třeba nejprve stáhnout ze zdroje. Protože je balíček zabalen je třeba jej nejprve rozbalit a následně zkonfigurovat:

```
wget http://www.tortall.net/projects/yasm/releases/yasm-1.2.0.tar.gz
tar -xvzf yasm-1.2.0.tar.gz
cd yasm-1.2.0
./configure & make & make install
```

Následně je třeba doinstalovat podporu pro audiokodek OPUS, který je určen pro WebRTC hovory. Instalace je prováděna obdobně jako u předchozího balíčku, jen p59kay configure je prováděn s dodatečnými parametry:

```
wget http://downloads.xiph.org/releases/opus/opus-1.1.tar.gz
./configure --with-pic --enable-float-approx
```

3.1.2 Ffmpeg

Následující balíček Doubango potřebuje pro své fungování další hlavičky a knihovny. U některých tipů ubuntu je nainstalována knihovna libav, která je velmi podobná knihovně ffmpeg. Tuto knihovnu je však třeba odinstalovat pro správné zkompilování a fungování ffmpeg knihovny pomocí příkazu:

```
apt-get remove libavutil51
```

Nyní můžeme přistoupit ke kompilaci samotného balíčku ffmpeg. Nejdříve je třeba stáhnout balíček z uložiště a následně jej zkompilovat s jednotlivými povolenými a zakázanými parametry, které jsou určeny pro následné správné fungování balíčku doubango. Všechny balíčky prvotně stahujeme do adresáře `/usr/local/src` a kompilace probíhá pomocí příkazů:

```
wget -c http://ffmpeg.org/releases/ffmpeg-1.0.2.tar.gz
```

```
./configure --extra-cflags="-fPIC" --extra-ldflags="-lpthread" --
enable-pic --enable-memalign-hack --enable-shared --disable-static -
-disable-network --disable-protocols --disable-pthreads --disable-
devices --disable-filters --disable-bsfs --disable-muxers --disable-
demuxers --disable-parsers --disable-hwaccels --disable-ffmpeg --
disable-ffplay --disable-ffserver --disable-encoders --disable-
decoders --disable-zlib --enable-gpl --disable-debug --enable-
encoder=h263 --enable-encoder=h263p --enable-decoder=h263 --enable-
encoder=mpeg4 --enable-decoder=mpeg4 --enable-libx264 --enable-
encoder=libx264 --enable-decoder=h264
```

3.1.3 **Doubango**

Doubango je IMS framework a obsahuje všechny signalizační protokoly (SIP, SDP, WebSocket, atd.) a specifikace pro přenos médií (RTP stack, audio/video codecs...) potřebné pro webrtc2sip gateway.

```
svn co http://doubango.googlecode.com/svn/branches/2.0/doubango
doubango
```

Následně bylo třeba vytvořit skript autogen.sh, který je třeba pro kompilaci balíčku doubango pomocí příkazu:

```
sed -i '1,/==/s/==/=/' autogen.sh
```

Pro úspěšné spuštění skriptu je potřeba doinstalovat balíčky pro příkazy, které jsou ve skriptu obsaženy:

```
apt-get install build-essential automake dh-autoreconf autoconf2.13
```

Nyní již příkaz `./autogen.sh` pro spuštění skriptu proběhl bez problémů a bylo možné dále postupovat v kompilaci se zadanými parametry:

```
./configure --with-ssl --with-srtp --with-vpx --with-speex --with-
speexdsp --enable-speexresampler --enable-speexjb --enable-
speexdenoiser --with-ffmpeg --with-h264 -prefix=/usr/local
```

Po úspěšné instalaci balíčku doubango byly v konzoli zobrazeny nastavené parametry viditelné na obrázku 3.2. Toto nastavení bylo provedeno pomocí dodatečných parametrů u příkazu `./configure`.

```
FFmpeg:                yes
VP8 video codec:       yes
OpenH264 video codec:  no
OPUS audio codec:      no
ILBC audio codec:      no
G.729 audio codec:     no -> yes
GSM audio codec:       no
AMR audio codec:       no -> yes
SPEEX audio codec:     yes
G.722 audio codec:     yes
G.711 audio codec:     yes

YUV:                   no
JPEG:                  no
SPEEX DSP:             yes

SSL:                   yes
DTLS-SRTP:             yes
DTLS:                  yes

SRTP:                  yes

WebRTC:                Enabled(no): AEC(no), NS(no)
```

Obrázek 3.2: *Úspěšné zkompilování balíčku doubango*

3.1.4 Webrtc2sip

Webrtc2sip je brána využívající prvků WebRTC a SIP signalizace pro vytvoření možnosti volání z webového prohlížeče. Je určen především pro překlady jednotlivých audio a video kodeků [19]. Brána webrtc2sip se skládá z několika modulů:

- **SIP Proxy** – je určena pro konverzi SIP z WebSocket paketů do paketů protokolů UDP, nebo TCP. Tato konverze je prováděna z důvodu lepší podpory protokolů UDP a TCP v externích sítích.
- **RTCWeb Breaker** – ve specifikaci WebRTC je definovaná povinná podpora pro ICE a DTLS/SRTP. Tento modul je určen pro konvertování a vyjednávání datového toku médií, aby byla možná komunikace s koncovými SIP zařízeními, které nepodporují definované protokoly.
- **Media Coder** - v WebRTC jsou definovány dva typy audiokodeků MTI (Mandatory To Implement – povinné k implementaci) a to opus a g.711. A dva videokodeky MTI, a to VP8 a H.264.
- **Click-to-Call** – jedná se o službu, která umožňuje přijímat maily, sledovat například Facebook a hlavně vytvořit hovor na svůj mobilní pomocí jediného kliku.

Instalace

Bránu nainstalujeme obdobně jako balíček doubango, také je třeba spustit skript autogen.sh:

```
svn co http://webrtc2sip.googlecode.com/svn/trunk/ webrtc2sip
```

Při spouštění skriptu autogen.sh nastal problém s procházením adresářů, který je zobrazen na obrázku 3.3. Proto je nutné do MakeFile.am přidat na konec souboru následující řádek:

```
AUTOMAKE_OPTIONS = subdir-objects
```

```
automake: You are advised to start using 'subdir-objects' option throughout your
automake: project, to avoid future incompatibilities.
Makefile.am:44: warning: source file 'db/sqlite/sqlite3.c' is in a subdirectory,
Makefile.am:44: but option 'subdir-objects' is disabled
```

Obrázek 3.3: *Problém s užíváním sub directory*

Nyní proběhl autogen bez problému a je možné pokračovat dále:

```
./configure --with-doubango=/usr/local -- prefix=/usr/local
```

```
In file included from mp_net_transport.h:23:0,
                 from mp_c2c.h:23,
                 from mp_c2c.cc:19:
mp_object.h:23:23: fatal error: tsk_debug.h: No such file or directory
 #include "tsk_debug.h"
                       ^
compilation terminated.
```

Obrázek 3.4: *Chybějící soubor tsk_debug.h*

Zde nastal problém s chybějícím souborem tsk_debug.h viz obrázek 3.4, kde se jednalo o neúplnou předešlou instalaci. Proto jsem zdrojový kód tohoto souboru vyhledala na internetu a poté jej uložila do požadovaného umístění `/usr/local/src/doubango/` a dále bylo ještě nutné doinstalovat balíčky:

```
apt-get install libapr1-dev libssl1.0.0:i386
```

Dále je potřeba provést opět příkaz configure s parametry

```
./configure --with-doubango=/usr/local -- prefix=/usr/local LIBS=-
ldl
```

kterým vytvoříme správné vyhledání závislostí a následně využijeme příkaz `make` pro spuštění MakeFile a zkompilování balíčku.

3.1.5 Asterisk 11

Asterisk je úplná softwarová pobočková ústředna a poskytuje všechny potřebné funkce pro nastavení naší PBX. Verze 11 byla zvolena z důvodu podpory WebRTC. [20]

Před samotnou instalací doinstalujeme balíčky, které budeme potřebovat během procesu:

```
apt-get install ncurses-dev libsqlite3-dev
```

Následně postupujeme podobně jako v předešlých případech:

```
wget http://downloads.asterisk.org/pub/telephony/asterisk/ old-
releases/asterisk-11.1.2.tar.gz
```

Po úspěšné instalaci spustíme Asterisk a dostaneme se do příkazového řádku pomocí příkazu:

```
asterisk -rvvv
```

3.2 WebRTC s Asterisk 13

U této varianty je instalace mnohem jednodušší, není potřeba instalace tak velkého množství jednotlivých balíčků, ale je zde problém s dostupnou dokumentací, protože se jedná o relativně nové řešení. Před instalací samotného Asterisku 13 je třeba nainstalovat potřebné knihovny, především pro podporu SRTP protokolu:

```
libsrtp0
libsrtp-dev
```

3.2.1 Jansson

Pro správné zkompilování Asterisku 13 je nutné předem nainstalovat balíček Jasson, který je vyžadován pro fungování Asterisk 13. Balíček doinstalujeme pomocí:

```
wget http://www.digip.org/jansson/releases/jansson-2.5.tar.gz
./configure -prefix=/
```

3.2.2 Asterisk 13

Je to další nástupce PBX Asterisk, který je doplněn o nové možnosti nastavení a příkazy. Hlavní změnou je možnost využití konfiguračního souboru pjsip.conf místo sip.conf. Tato možnost byla již u Asterisk 12 a jedná se o nastavení pomocí jiného konfiguračního souboru s jinými typy parametrů. Hlavní výhodou je doplnění o další možnosti nastavení, v našem případě větší podpora pro WebRTC.[21]

Instalace balíčku je prováděna stejně jako u předešlých verzí:

```
wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-
13-current.tar.gz
./configure -with-crypto -with-ssl -with-srtp=/usr/local/lib
```

3.3 Generování klíčů a certifikátů

Pro fungování DTLS je třeba vygenerovat klíče a certifikáty pro zabezpečení. Klíče a certifikáty je nutné vygenerovat a využít u obou variant realizace. Následujícím postupem vytvoříme klíč a certifikát, kterým následně klíč podepíšeme:

```
# cd /usr/src/asterisk-*/contrib/scripts
# /ast_tls_cert -C pbx.mycompany.com -O "My Super Company" -d
/etc/asterisk/keys
# ./ast_tls_cert -m client -c /etc/asterisk/keys/ca.crt -k
/etc/asterisk/keys/ca.key -C phone1.mycompany.com -O "My Super
Company" -d /etc/asterisk/keys -o malcolm
```

3.4 Konfigurace PBX

Nastavení jednotlivých konfiguračních souborů je obdobné u obou variant instalace. Po instalaci Asterisku jsou tyto konfigurační soubory defaultně nastaveny v adresáři `/etc/asterisk` . Soubory je vhodné pouze přejmenovat a díky tomu si je uložit pro případné následné využití. Pro naše účely si vytvoříme nové konfigurační soubory.

sip.conf

```
[general]
udpbindaddr=0.0.0.0:5060
realm=158.196.244.143
transport=udp,ws
videosupport=yes
[6001]
host=dynamic
secret=6001
context=outgoing
type=peer
encryption=yes
avpf=yes
icesupport=yes
directmedia=no
disallow=all
allow=all
dtlsenable=yes
dtlsverify=fingerprint
dtlscertfile=/etc/asterisk/keys/asterisk.pem
```

```
dtlscafile=/etc/asterisk/keys/ca.crt
dtlssetup=actpass
```

V části konfiguračního souboru pojmenované general, nastavujeme parametry, které jsou společné pro všechny klienty uvedené v sip.conf. Nastavením udpbindaddr na IP adresu 0.0.0.0 bude PBX přijímat zprávy a data ze všech IP adres na portu 5060. Realm je IP adresa serveru kde je nastavena PBX. Pro přenos využíváme protokoly WebSocket a UDP. Povolením podpory videa zaručujeme podporu videohovoru. Při nastavení klienta volíme ty dynamic, tímto umožníme připojení klienta z jakékoliv IP adresy. Po nastavení hesla nastavíme tip klienty. Parametr peer určuje, že komunikace může probíhat jen mezi dvěma IP adresami. V položce kontext přiřadíme daný kontext z diaplanu, v našem případě s názvem outgoing. Při povolení avpf nastavíme interoperabilitu pro mediální streamy pomocí profilu AVPF RTP. Povolíme-li encryption, tak je možné data zapouzdřit pomocí protokolu SRTP. Parametrem icesupport je povoleno překonání NAT a directmedia no určuje, že i média musejí jít přes PBX a není možné je posílat přímo mezi klienty. Parametry začínajícími na dtls nastavuje DTLS zabezpečení provozu. Nejprve jej povolíme a poté zadáme cestu k certifikátu a k certifikátu certifikační autority. Nastavením setup na aktivní určujeme parametrem actpass, který říká, že se chceme spojit s protější stranou zabezpečení.

http.conf

```
[general]
enabled=yes
bindaddr=0.0.0.0
bindport=8088
```

U konfiguračního souboru http.conf nastavíme podobně jako u předešlého sip.conf, takže budeme přijímat od všech IP adres na portu 8088.

rtp.conf

```
 [general]
rtpstart=10000
rtpend=20000
icesupport=yes
stunaddr=stun.l.google.com:19302
```

Parametry start a end nastavíme rozsah jednotlivých portů. Jako STUN server využijeme server společnosti Google, který by měl být funkční a dostupný vždy a odkudkoliv.

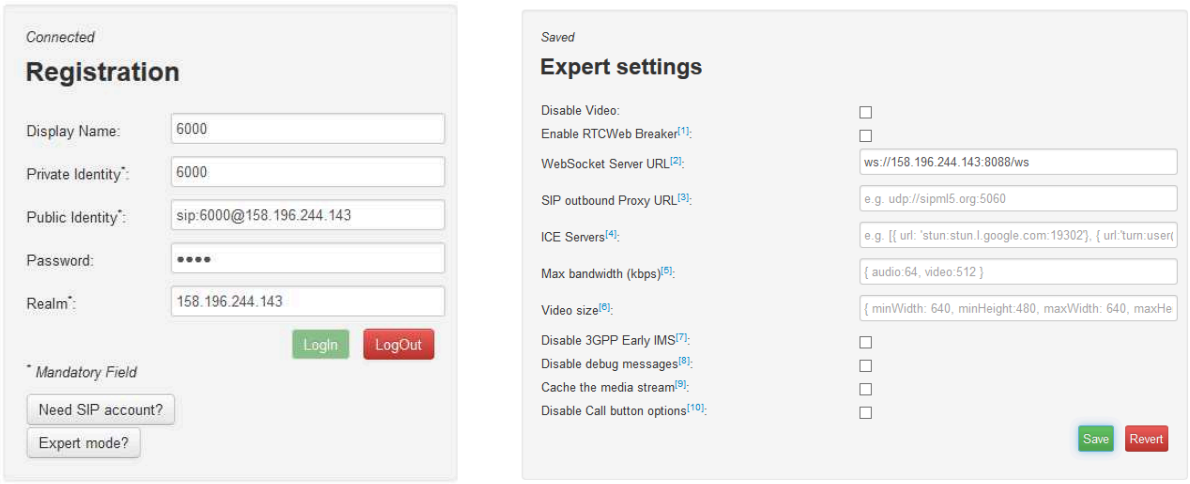
extensions.conf

```
[outgoing]
exten => _X.,1,Dial(SIP/${EXTEN})
exten => _X.,n,Hangup()
```

V dialplanu nastavíme scénář, že na zvolené číslo realizujeme hovor a po ukončení se zavěsí a hovor se ukončí. Parametr EXTEN nám zastupuje hodnotu vytočeného čísla. Podobně jako parametr X, který určuje, že číslo je stejné jako o řádek výše, což ulehčuje práci při sestavování diaplanu.

3.5 Nastavení volání pomocí SipML5 klienta

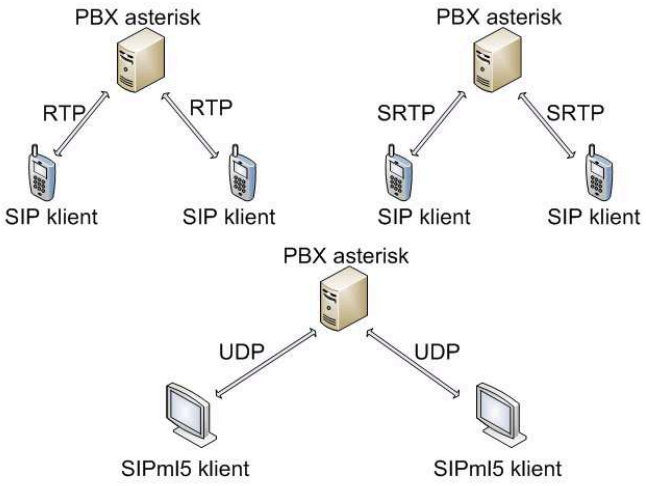
Na webové stránce http://sipml5.org/call.htm nastavíme jednotlivé parametry (viz obrázek 3.5) pro oba webové klienty. Každého klienta nastavíme na jiném PC a uskutečníme hovor.



Obrázek 3.5: *Nastavení SipML5 klienta*

3.6 Vytvoření konvenčních SIP relací

V další části své práce jsem sestavila nová dvě pracoviště pro měření. Jedná se o klasickou SIP relaci, kde je pro přenos audia nebo videa využíváme protokol RTP. Další měřený přenos, byl přenos dat pomocí protokolu SRTP. Na pracovištích viditelných na obrázku 3.6 jsem měřila přenos audio a video streamu.



Obrázek 3.6: *Jednotlivé pracoviště pro měření*

Sestavení pracovišť probíhá podobně jako u realizace WebRTC. Parametry však měníme pouze u SIP nastavení a dialplanu, ostatní konfigurační soubory zůstávají v základním nastavení. Diaplan můžeme využít stejný jako v předešlém případě. Změny se budou týkat hlavně sip.conf, kde již není potřeba povolovat DTLS, avpf a nastavovat příslušné parametry. Pro naše potřeby se změní například i transportní protokoly a další položky. V jednotlivých případech je nutné povolit podporu videa pro realizaci videohovorů a také encryption pro přenos pomocí protokolu SRTP.

Abych zachytila provoz v obou směrech, prováděla jsem zachytávání provozu na počítači s PBX Asterisk. Zachytávání bylo realizování pomocí nástroje tcpdump, kde příkaz pro spouštění monitorování je ve tvaru:

```
tcpdump –S –i eth0 –w název-výstupu.pcap
```

Zachycený provoz jsem poté zpracovávala pomocí software wireshark, kde jsem získala potřebná data pro následnou analýzu.

4 Porovnání WebRTC scénářů s konvenčními SIP relacemi

V poslední části své diplomové práce jsem vyhodnocovala dříve naměřená data. Tyto data jsem porovnávala z pohledu zatížení na síť. V případě audio streamu jsem ještě vypočítala teoretické hodnoty a porovnala je s hodnotami naměřenými na jednotlivých pracovištích.

4.1 Výpočet teoretických hodnot a očekávaných bitových toků

Pro možnost porovnat naměřená data s hodnotami teoretickými je třeba nejdříve tyto hodnoty vypočítat. Teoretické hodnoty lze vypočítat pouze u audio provozu, protože video pakety mají různé délky dle typu snímku, proto není možné je vyjádřit vzorcem [15]. Vzorce pro teoretické hodnoty jsou následující:

$$\Delta t = \frac{P_S}{C_R} \tag{4.1}$$

$$B_{W_M} = M \cdot \left(\frac{\sum H + P_S}{\Delta t} \right) \tag{4.2}$$

Ze vzorce 4.1 a 4.2 lze odvodit celkový vztah pro výpočet daný vzorcem 4.3:

$$B_{W_M} = M \cdot \left(\frac{\sum H + P_S}{\frac{P_S}{C_R}} \right) = M \cdot \left(\frac{\sum H + P_S}{P_S} \cdot C_R \right) = M \cdot C_R \cdot \left(1 + \frac{\sum H}{P_S} \right) \tag{4.3}$$

Kde

P_S…..… (Payload size) je velikost části paketu pro data [b]

C_R…… (Cipher rate) je rychlost proudu bitů z kodéru [kbit/s]

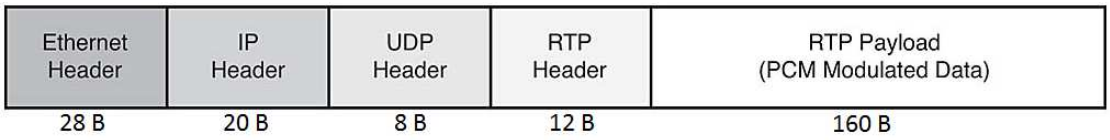
M……. Počet hovorů

∑ H..... Hodnoty jednotlivých hlaviček nacházejících se v paketu [b]

Ve své práci využívám provozu pomocí RTP, SRTP a UDP protokolů. Jako kodek jsem volila PCM g.711.U kterého je udávána C_R=64 kbit/s. Nyní pomocí parametrů a vzorců můžeme vypočítat teoretické hodnoty jednotlivých provozů.

4.1.1 Teoretická hodnota RTP

U RTP paketů musíme počítat se všemi hlavičkami, co se v paketu nacházejí, jak je viditelné na obrázku 4.1. Součástí paketu je Ethernet, IP, UDP a RTP hlavička a jejich jednotlivé velikosti. Velikosti hlaviček jsou uváděny v bajtech. My je však pro výpočet převedeme na bity, aby všechny jednotky ve vzorci byly shodné.



Obrázek 4.1: *RTP Paket*

H_{UDP}=8B =64 b

HIP=20B =160 b

HETH=28B =224 b

H_{RTP}=12B = 96 b

P_S=160 B = 1280 b

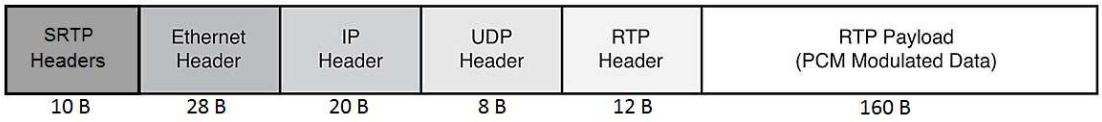
C_R=64 kbit/s

$$\Delta t = \frac{P_S}{C_R} = \frac{1280}{64} = 20ms$$

$$B_{W_M} = M \cdot \left(\frac{\sum H + P_S}{\Delta t} \right) = 1 \cdot \left(\frac{224 + 160 + 64 + 96 + 1280}{20} \right) = 91{,}2kbit/s$$

4.1.2 Teoretická hodnota SRTP

U SRTP paketů musíme připočítat ještě SRTP hlavičky. U těchto hlaviček jsou dvě možnosti velikosti, protože hlavičky se zde nacházejí dvě, kdy jedna je volitelná. Varianta velikostí je 10 B a 4 B. My budeme počítat s 10 B, protože naměřené pakety mají hlavičku 10 B.



Obrázek 4.2: *SRTP paket*

H_{UDP}=8B =64 b

HIP=20B =160 b

HETH=28B =224 b

P_S=160 B = 1280 b

C_R=64 kbit/s

H_{RTP}=12B = 96 b

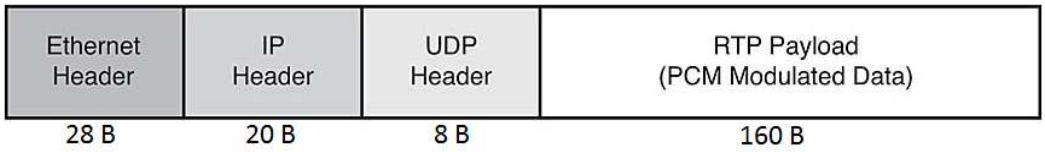
H_{SRTP}=10B = 80 b

$$\Delta t = \frac{P_S}{C_R} = \frac{1280}{64} = 20ms$$

$$B_{W_M} = M \cdot \left(\frac{\sum H + P_S}{\Delta t} \right) = 1 \cdot \left(\frac{80 + 224 + 160 + 64 + 96 + 1280}{20} \right) = 95{,}2kbit / s$$

4.1.3 Teoretická hodnota UDP

U WebRTC jsou data přenášena pomocí UDP protokolu. Tento paket obsahuje méně hlaviček, než předešlé dva typy paketů, takže bude potřeba méně prostoru pro přenos. Velikosti jednotlivých částí paketu jsou znázorněny na obrázku.



Obrázek 4.3: *UDP paket*

H_{UDP}=8B =64 b

HIP=20B =160 b

HETH=28B =224 b

P_S=160 B = 1280 b

C_R=64 kbit/s

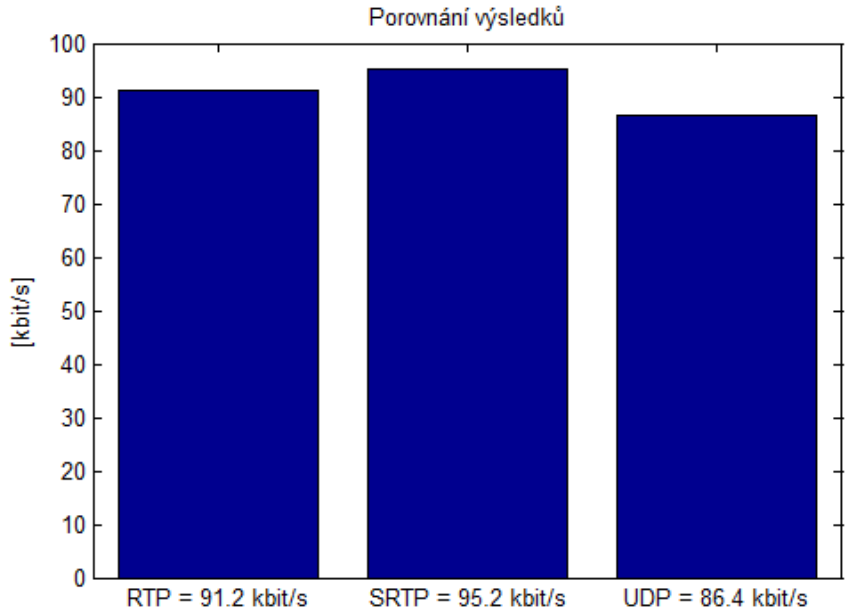
$$\Delta t = \frac{P_S}{C_R} = \frac{1280}{64} = 20ms$$

$$B_{W_M} = M \cdot \left(\frac{\sum H + P_S}{\Delta t} \right) = 1 \cdot \left(\frac{80 + 224 + 160 + 64 + 1280}{20} \right) = 86{,}4kbit / s$$

Jednotlivé teoretické hodnoty odpovídají předpokladům, že čím je větší paket, tím jsou větší nároky na přenos. Velikost paketu ovlivňuje především míra zabezpečení a dalších mechanismů. Pro přehled rozdílů mezi jednotlivými protokoly jsem výsledky uvedla do tabulky 4.1 a do grafu na obrázku 4.4.

Tabulka 4.1: *Výsledky jednotlivých výpočtů*

Protokol	**Nároky na přenos**
RTP	91,2 kbit/s
SRTP	95,2 kbit/s
UDP	84,6 kbit-s



Obrázek 4.4: *Výsledky jednotlivých teoretických výpočtů*

4.2 Získání dat z reálných experimentů

U porovnávaných paketů RTP, SRTP i UDP je výhodou, že jsou v reálném čase situovány za sebou, proto nemusíme brát v potaz ostatní pakety v souboru a můžeme si vyfiltrovat pouze požadované pakety.

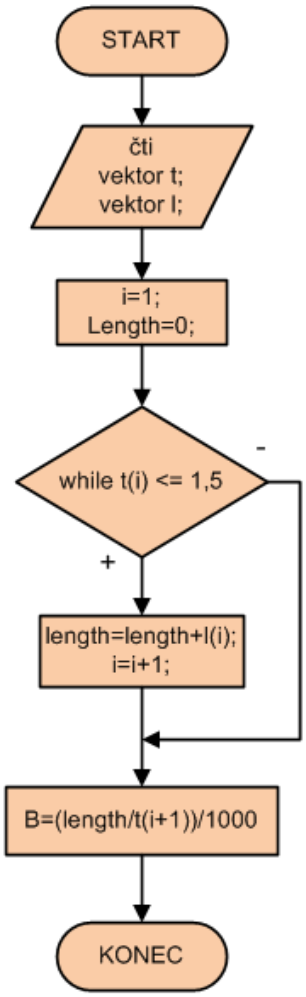
```
rtp and ip.src==‟zdrojova IP‟ and ip.dst==‟cilova IP‟
```

Filtrování provádíme pro daný protokol pro každý směr provozu zvlášť. Po vyfiltrování využijeme funkce software wireshark, který nám umožňuje data uložit v textové podobě. Využijeme možnosti „Export packet Dissections as “CSV“ (Coma Separated Values packet summary) file“ pro získání vhodného textového výstupu. Při vygenerování pouze potřebných paketů docílíme nastavení času u prvního paketu na hodnotu nula, což nám velmi zjednodušuje následnou manipulaci s daty.

Tento soubor následně otevřeme v tabulkovém editoru a můžeme získanou tabulku jednoduše upravovat, například smazat nepotřebné sloupce. Pro naše účely potřebujeme jen sloupce „Time“ a „Lenght“. Přidáme další sloupec, ve kterém přepočítáme velikosti paketů na bity. Tuto úpravu provádíme z důvodu dalších výpočtů, které jsou počítány v bitech. Další nutnou úpravou je přičtení 14 B (112 b) k hodnotě RTP a SRTP paketů. Tato úprava je nutná z důvodu nepřesných velikostí paketů udávaných software wireshark, který nepočítá se všemi hlavičkami obsaženými v paketu.

4.3 Nástroj na zpracování dat

Pro každý druh provozu vytvoříme jednotlivé skripty. Nejdříve spočítáme teoretickou hodnotu, kterou budeme následně porovnávat s hodnotami naměřenými. Vyhodnocení naměřených hodnot probíhá pomocí schématu viditelného na obrázku 4.5. Velikosti jednotlivých paketů a časy trvání paketů se sčítají a dosáhne-li velikost časové hodnoty větší nebo rovné 1,5 máme již sečtené dostatečné množství dat pro vyhodnocení. Součet jednotlivých velikostí paketů následně podělíme výsledným časem, abychom získali velikost paketů přenesených za jednu sekundu. Tuto hodnotu ještě podělíme tisícem, aby výsledná jednotka byla v jednotkách kbit/s.



Obrázek 4.5: *Vývojový diagram skriptu*

Tyto vyhodnocení provedeme pro každý směr provozu zvlášť a výsledné hodnoty následně porovnáme v grafu s teoretickou vypočítanou hodnotou.

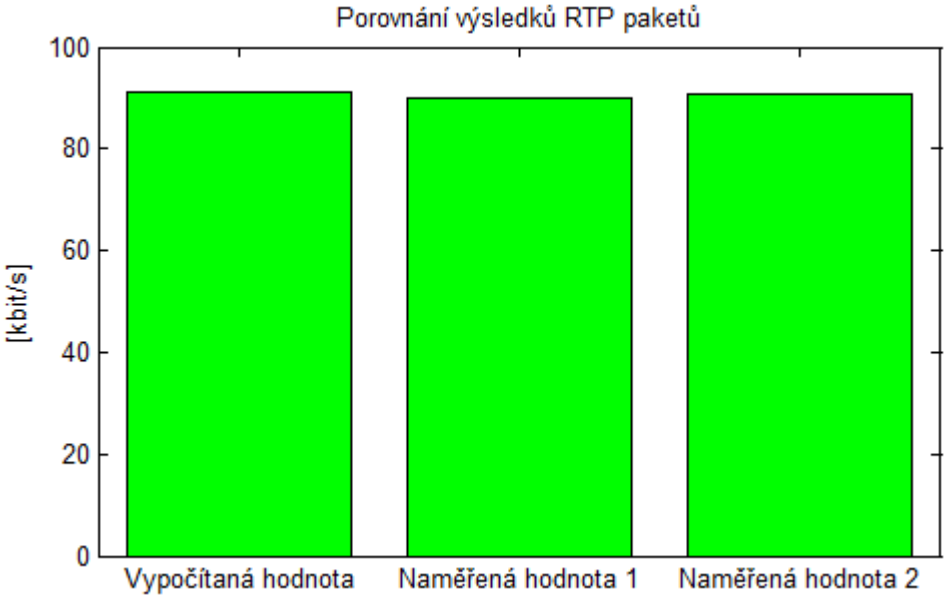
Teoretické výpočty a analýza naměřených dat je prováděna pomocí software matlab[22]. Vypočítané a naměřené hodnoty jsou následně znázorněny v grafu. Zdrojový kód je znázorněn v příloze A, kde je vidět zpracování jednotlivých sad vzorků a následné vynesení do grafu.

4.4 Porovnání měření audio paketů a teoretických výpočtů

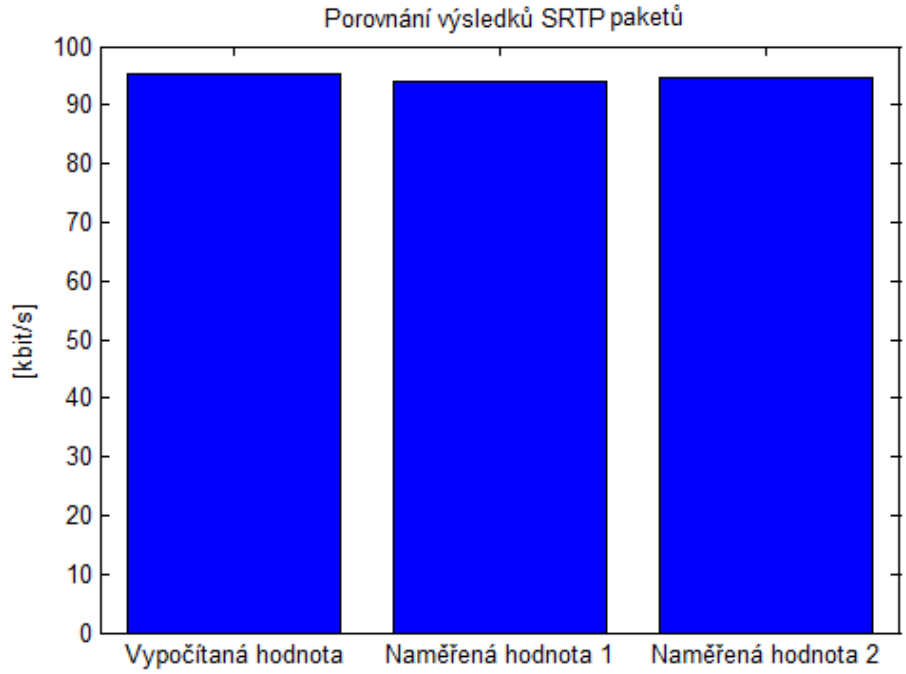
U analýzy audio paketů jsem volila jedno měření a následně jsem provoz rozdělila podle směrů provozu na další dvě měření. Díky rozdělení podle směrů jsem získala dvě sady hodnot na zpracování. U audio paketů je dostačující menší množství měření, protože naměřené hodnoty jsou velmi blízké předpokládanému modelu, jak je možno vidět v tabulce 4.2. a na jednotlivých výsledných grafech (obr. 4.6, 4.7, 4.8).

Tabulka 4.2: *Výsledné hodnoty měření a výpočtů*

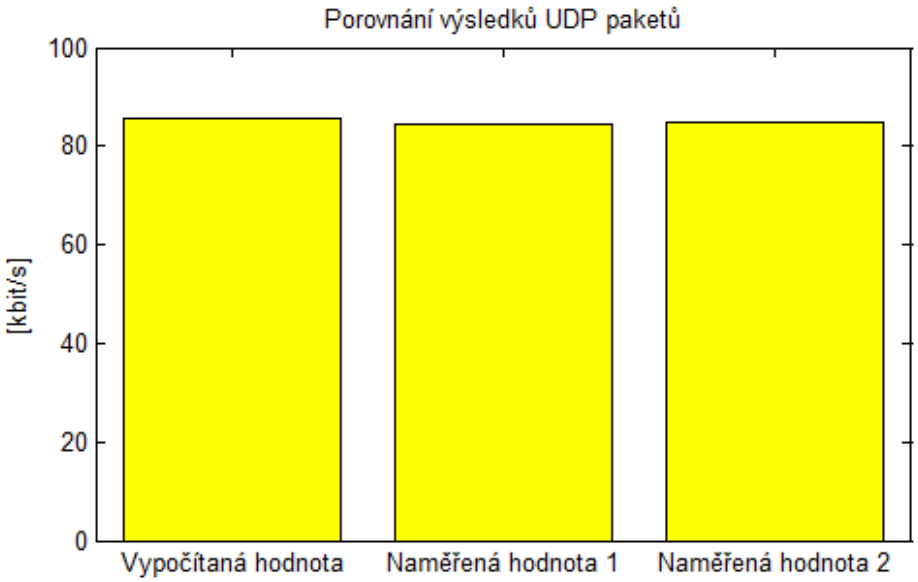
Protokol	**Vypočítaná hodnota**	**Naměřená hodnota downlink**	**Naměřená hodnota uplink**
RTP	91,2000 kbit/s	90,0323 kbit/s	90,6207 kbit/s
SRTP	95,2000 kbit/s	94,0551 kbit/s	94,5919 kbit/s
UDP	85,6000 kbit/s	84,4721 kbit/s	84,9568 kbit/s



Obrázek 4.6: *Graf výsledných hodnot RTP*



Obrázek 4.7: *Graf výsledných hodnot SRTP*



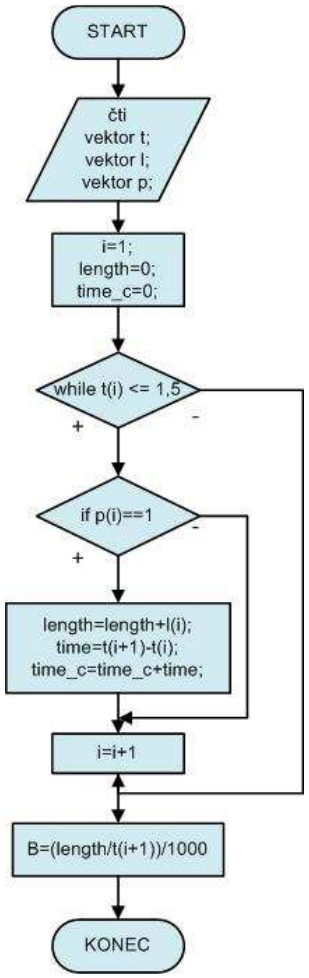
Obrázek 4.8: *Graf výsledných hodnot UDP*

Odchylka naměřených hodnot je minimální od teoretických hodnot vypočítaných. U jednotlivých měření je odchylka v řádu 1,2 kbit/s u download směru a 0,7 kbit/s u uplink směru. Díky těmto malým odchylkám můžu říci, že je ověřena validita jednotlivých výpočetních vztahů pro nároky audia na přenos v IP sítích.

4.5 Výsledky analýzy video paketů

U video paketů je délka paketů variabilní, protože přenášíme různé objemy dat. Pro vyhodnocení je třeba získání dostatečně velkého množství dat, proto budeme využívat všechny naměřené hodnoty. Z toho důvodu musíme v tabulce vytvořit konec pro vyhledávání souboru. Já jsem volila číslo nula, které se v daném sloupci nikde jinde neobjevuje a díky rozeznání nuly se cyklus ukončí. Při zachycení celého postupu hovoru jsou video pakety prokládány zprávami SIP, v případě WebRTC jsou video pakety prokládány STUN a DTLS pakety, se kterými také musíme počítat při výpočtech časů paketů. Proto je program pro vyhodnocení poněkud odlišný. Využijeme indikátoru typu protokolu, který byl vytvořen v tabulkovém editoru pomocí podmínky if, kde vyhledáváme používaný přenosový protokol.

Podmínka říká, že když v daném poli bude hodnota „UDP“, bude zde vložena jednička, v opačném případě nula. Tyto rozhodovací hodnoty následně využijeme při výpočtech v programu matlab. Program musíme upravit, aby se sčítaly pouze pakety, které vyhodnocujeme. Zároveň potřebujeme i ostatní pakety pro správné vyhodnocení délky trvání paketu. Vývojový diagram procesu bude vypadat následovně:



Obrázek 4.9: *Vývojový diagram pro zpracování video paketů*

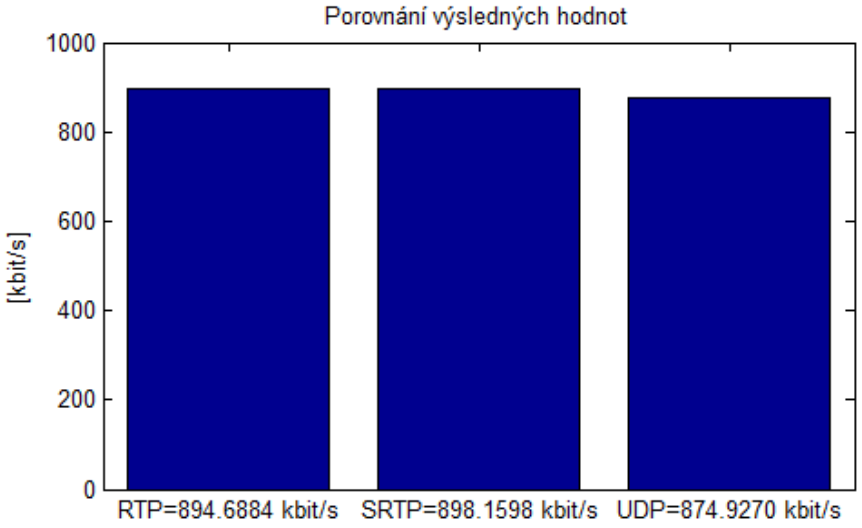
kde vektor p je indikátor protokolu a jsou sčítány pouze pakety vybraného protokolu. Dobu pro přenos paketu počítáme jako rozdíl času paketu následujícího a paketu, který vyhodnocujeme. Program v software maltab bude vypadat obdobně, jen cyklus bude pozměněn o podmínku UDP paketů. Pozměněný zdrojový kód je uveden v příloze B.

U každého typu provozu jsem provedla tři měření, a protože vyhodnocuji každý směr zvlášť, mám ve výsledku šest skupin dat pro analýzu. Provedla jsem analýzu jednotlivých naměřených dat a z výsledných hodnot vytvořila průměrnou hodnotu. Měření bylo prováděno pomocí standardu H.264 pro rozlišení videa 640 x 480 pixelů. Jednotlivé výsledky jsem vynesla do tabulky 4.3 a grafů (obr. 4.10 – 4.13).

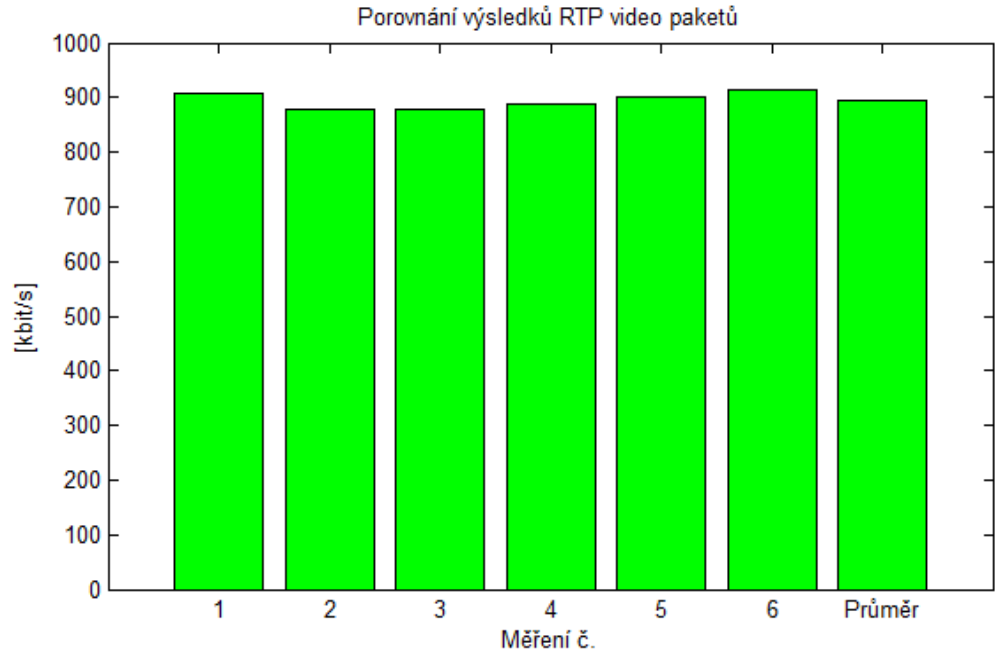
Tabulka 4.3: *Jednotlivé výsledky měření*

	Měření č.1		**Měření č.2**		**Měření č.3**		[kbit/s]
Protokol	**1-down**	**2-up**	**3-up**	**4-down**	**5-up**	**6-down**	**Průměr**
RTP	908,8083	878,6415	878,6417	886,7323	901,7456	913,5610	894,6884
SRTP	908,1815	896,8818	887,0557	895,0001	904,0959	897,7442	898,1598
UDP	902,5652	893,3863	849,5239	849,5241	872,7663	881,7963	874,9270

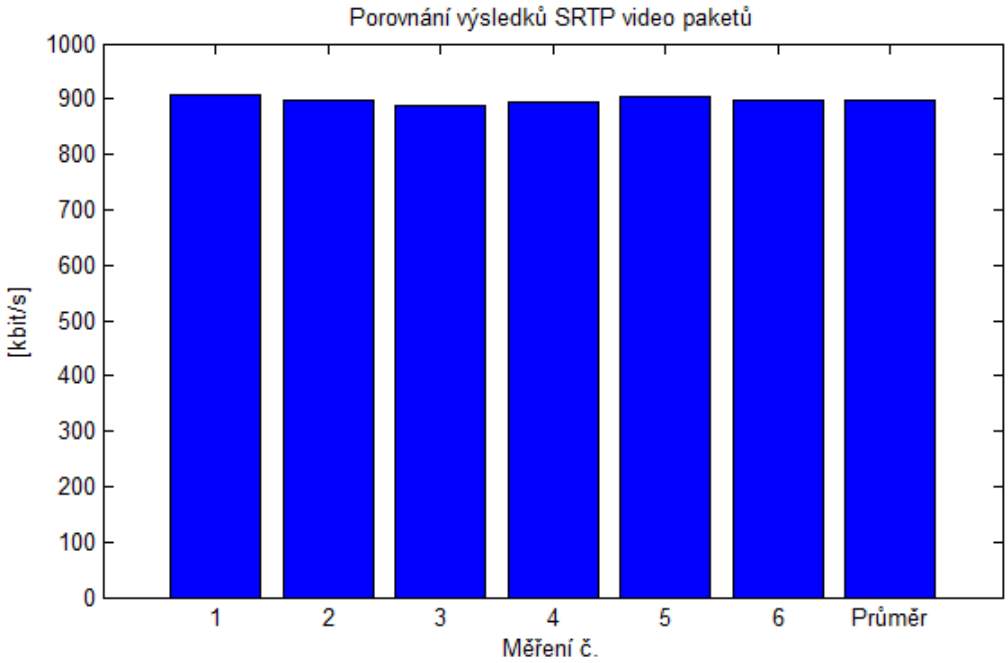
Z jednotlivých měření vidíme, že odchylky měření jsou minimální, a proto není třeba provádět více měření. Výsledné průměrné hodnoty by se i při provedení většího počtu měření lišily jen velmi málo.



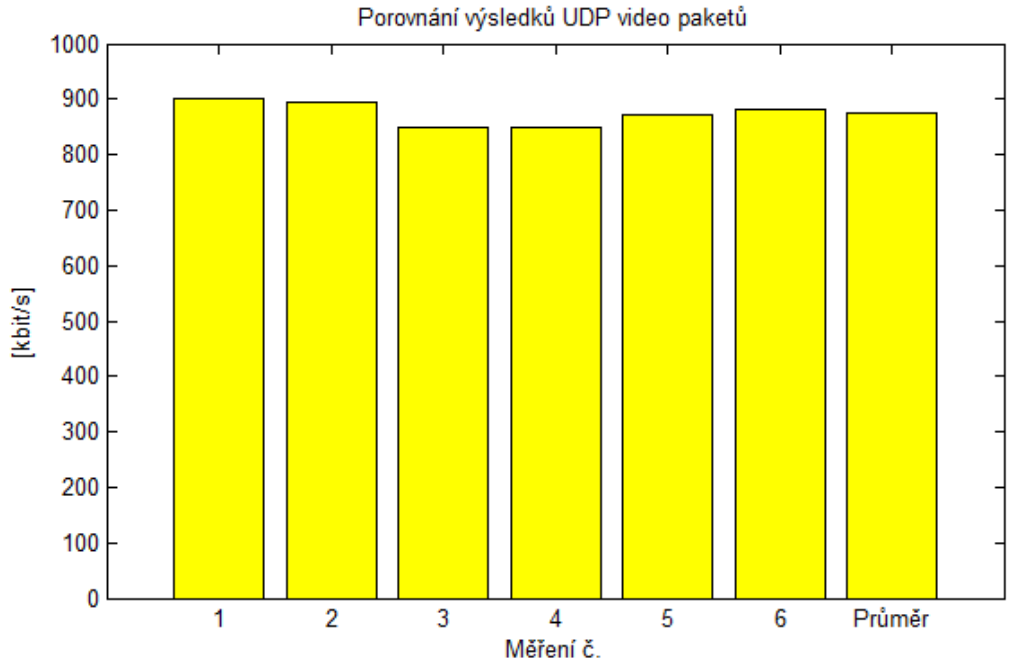
Obrázek 4.10: *Graf porovnání výsledných průměrných hodnot*



Obrázek 4.11: *Graf výsledků měření protokolu RTP*



Obrázek 4.12: *Graf výsledků měření protokolu SRTP*



Obrázek 4.13: *Graf výsledků měření protokolu UDP*

4.6 Program pro výpočet celkových nároků na provoz

V závěru své práce jsem vytvořila nástroj pro výpočet více hovorů najednou. Uživatel je vyzván k zadání počtu audio a video hovorů a pro volbu přenosového protokolu. Následně je počtem hovorů vynásobena průměrná hodnota získaná v jednotlivých měřeních. Výběr protokolu pro přenos je řešen pomocí výběru jednotlivých položek cyklu switch, kde uživatel je vyzván, aby zadal hodnotu od 1 do 4 pro zadaný protokol. Vyzvání uživatele pro zadání vstupu je realizováno pomocí příkazu:

```
pocet = input('zadejte počet hovorů: ');
```

kde zadaná hodnota je následně uložena do proměnné a je možné s ní dále pracovat. Výstupem z programu je výsledná hodnota nároků na přenos pro hovory zadaných parametrů. Skript programu je uveden v příloze C.

4.6.1 Příklad práce s programem

Uživatel je vyzván k zadání jednotlivých parametrů, kterými jsou počet audio a video hovorů a volba protokolu pro přenos streamu:

```
Zadejte počet audio hovorů: 5
Zadejte počet video hovorů: 8
Vyberte protokol pro přenos (RTP=1, SRTP=2, UDP=3, všechny=4): 1
```

Následně se uživateli zobrazí jeho zvolené parametry a je proveden výpočet. Příkladem jsou vypočítány nároky na přenos pěti audio hovorů a osmi video hovorů. Jako protokol pro přenos byl zvolen RTP:

```
Zvolili jste 5 audio hovorů a 8 videohovorů
Zvolený kodek pro přenos je RTP
Celkové nároky na přenos jsou: 7609.139700 kbit/s
```

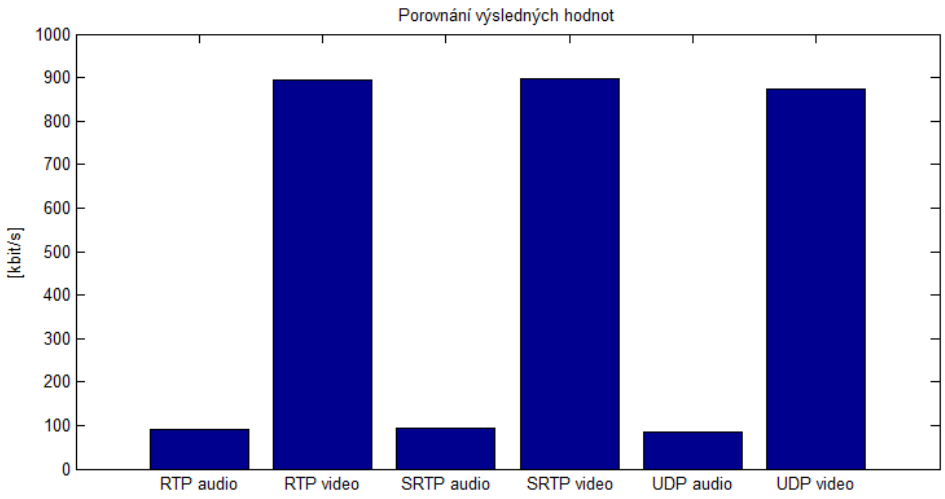
V případě že uživatel zvolí pro přenos všechny dostupné protokoly, vypočítají a zobrazí se výsledné hodnoty pro všechny zadané parametry. V tomto případě se jednalo o osm audio hovorů a šest video hovorů:

```
Zvolili jste 8 audio hovorů a 6 videohovorů
Zvolili jste všechny typy kodeků
Celkové nároky na přenos pomocí RTP jsou: 6090.742400 kbit/s
Celkové nároky na přenos pomocí SRTP jsou: 6143.546800 kbit/s
Celkové nároky na přenos pomocí UDP jsou: 5927.278000 kbit/s
```

5 Zhodnocení

Ve své práci jsem porovnávala naměřená data s teoretickými výpočty při přenosu audio streamu. Měření bylo prováděno na reálném hovoru mezi dvěma SIP klienty pomocí Asterisk pobočkové ústředny a mezi dvěma sipML5 klienty, mezi kterými probíhala komunikace pomocí Asterisk ústředny nastavené na požadavky WebRTC. Naměřené hodnoty jsem vyhodnocovala pomocí software matlab, kterým jsem prováděla postupné sčítání paketů a výslednou délku poté vztáhla vzhledem k času. Naměřené hodnoty byly velmi blízké hodnotám teoretickým s odchylkou maximálně 2 kbit/s v obou směrech, ať už uplink nebo downlink. Díky tomuto měření jsem si ověřila, že vzorce pro teoretické předpoklady jsou správné. V měření jsem si také ověřila předpoklad, že protokol, jehož paket obsahuje největší množství jednotlivých hlaviček, které navyšují délku paketu, má největší nároky na přenos v síti.

U video streamu bylo měření poněkud složitější, protože pakety přenášející video stream jsou variabilní velikosti. Rozdílnost velikosti paketů je dána principem kódování videa pro přenos, kde jsou kódovány hlavně rozdíly mezi jednotlivými snímky. Takže by se dalo říci, že pokud bude volající u hovoru pouze sedět, budou rozdíly minimální a tím i velikost paketů. Pokud bude ale volající aktivní, co se týče pohybu, budou rozdíly velké a velikost paketů bude velmi rozdílná. Měření bylo prováděno obdobně jako u audio paketů s tím rozdílem, že bylo provedeno více měření. Provedla jsem tři měření a po rozdělení provozu do jednotlivých směrů jsem získala šest sad vzorků pro analýzu. Prováděla jsem více měření z důvodu variability velikosti paketů a z výsledků jednotlivých analýz jsem vytvořila průměrnou hodnotu. Výsledné hodnoty nároků na video pro přenos jsou ale velice nízké, což bylo způsobeno rozlišením kamer na jednotlivých zařízeních. Vždy jeden klient byl realizován na mobilním zařízení, u kterého nebyla kamera moc kvalitní. Jednotlivé výsledky měření jsem pro porovnání vynesla do grafu znázorněného na obrázku 5.1.



Obrázek 5.1: *Zhodnocení výsledků měření*

Závěr

V teoretické části své práce jsem popsala vlastnosti jednotlivých audio a video standardů, které jsou podporovány technologií WebRTC. Jedná se především o audio standardy iLBC, ISA a OPUS. U video standardů jsou uvedeny standardy VP8 a H.264, ale také jejich nástupci VP9 a H.265. Také jsem zmínila jednotlivé platformy pro podporu multimédií ve webovém prostředí, jako jsou Adobe Flash Player a Microsoft Silverlight. V další části je vysvětlena architektura technologie WebRTC a především směrovací protokol SIP a jeho vlastnosti.

V praktické části své práce jsem nejprve sestavila spojení WebRTC technologie, která byla realizována pomocí Asterisk pobočkové ústředny a dvou sipML5 klientů. Klienti byli realizováni ve webovém prohlížeči pomocí již předem vytvořeného nástroje společnosti Doubango Telecom. Obsluha tohoto nástroje je velmi jednoduchá a výhodou je, že uživatelské rozhraní a nastavení je vždy stejné, takže odpadá problém s rozdílností konfigurací, která se objevuje u SIP klientů. Nevýhodou technologie WebRTC je malé množství dokumentace pro konfiguraci pomocí nástroje Asterisk. K této technologii jsou dostupné jednotlivé publikace, ale ty řeší WebRTC z pohledu vývoje webového rozhraní a mechanismu provozu WebRTC klientů. Realizace pomocí nástroje Asterisk je dokumentována pouze na jednotlivých webových návodech nebo v odborných fórech na internetu.

V další části jsem sestavila pracoviště pro volání mezi SIP klienty, kde je využíváno pro přenos audio i video streamu protokolu RTP a SRTP. Na těchto pracovištích a na WebRTC pracovišti jsem provedla jednotlivá měření, abych získala data pro následnou analýzu. Analýza dat byla prováděna z provozu zachyceného při volání mezi SIP a sipML5 klienty. Naměřená data byla následně zpracovávána v software matlab. U audio provozu jsem naměřené hodnoty porovnávala s předem vypočítanými teoretickými hodnotami pro nároky na přenos v síti. Naměřené hodnoty a hodnoty teoretické byly téměř totožné s velmi malou odchylkou, takže jsem si ověřila, že měření a výpočty byly správné. U video provozu nelze nároky vypočítat z důvodu různých velikostí paketů přenášející tento video stream. Proto jsem provedla větší množství měření a následně jsem data zprůměrovala.

Výsledné hodnoty jsem vynesla do tabulek a grafů pro porovnání. Naměřená data jsem ještě následně využila v nástroji pro traffic engineering, který vypočítá výsledné nároky na síť pro jednotlivé zadané parametry. Při využití nástroje si uživatel zvolí počet audio a video hovorů, a také typ protokolu pro přenos.

Výsledkem mé práce jsou analýzy jednotlivých druhů přenosu a v případě audio provozu porovnání s hodnotami teoretickými. Dále pak využití výsledných hodnot pro vytvoření traffic engineering nástroje pro výpočet celkových nároků na přenos.

Použitá literatura

[1] IETF RFC 6716. *Definition of the Opus Audio Codec*. 2012. Dostupné z: https://tools.ietf.org/html/rfc6716

[2] IETF. *RTP Payload Format for the iSAC Codec*. 2013. Dostupné z: https://tools.ietf.org/html/draft-ietf-avt-rtp-isac-04

[3] IETF. *RTP Payload Format for iLBC Speech*. 2004. Dostupné z: https://tools.ietf.org/html/draft-ietf-avt-rtp-ilbc-05

[4] RAIS, Ondřej. *H.264 (MPEG-4 part 10)*. [online]. 2004 [cit. 2015-05-03]. Dostupné z: http://www.golias.cz/index.php?modul=audio&sub=clanky&page=clanek&id=1697

[5] CHIP.CZ. *H.264: Jeden standard pro všechna videa*. [online]. roč. 2009, č. 04. [cit. 2015-05-03]. Dostupné z: http://www.chip.cz/soubory/dokumenty/afd9e4715dbac6b26fcdecb87cc67dfe.pdf

[6] CHIP.CZ. *H.265 špičkový videokodek*. [online]. roč. 2013, č. 07 [cit. 2015-05-03]. Dostupné z: http://www.chip.cz/soubory/dokumenty/07-13-108-spickovy-videokodek.pdf

[7] IETF RFC 6386. *VP8 Data Format and Decoding Guide*. Google Inc., 2011. Dostupné z: https://tools.ietf.org/html/rfc6386

[8] BANOSKI, Jim. *The VP8 video codec: High compression+low complexity*. 2009. Dostupné z: http://www.eetimes.com/document.asp?doc_id=1275667

[9] KLEJMOVA,Eva. *Srovnání výkonnosti nejnovějších standardů pro kódování video sekvencí*.2012. Bakalářská práce. Vysoké Učení Technické v Brně. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=54124

[10] IETF. *A VP9 Bitstream Overview*. Google Inc., 2013. Dostupné z: http://tools.ietf.org/html/draft-grange-vp9-bitstream-00

[11] MACDONALD, Matthew a Mario SZPUSZTA. *ASP.NET 3.5 a C# 2008*. Brno: Zoner Press, 2008. ISBN 978-80-7413-008-3.

[12] SUCHAN. *Microsoft Silverlight: Technologie, vlastnosti, uplatnění*. [online]. s. 16 [cit. 2015-05-03]. Dostupné z: http://ondrej.jikos.cz/vyuka/swi117/2008/microsoft-silverlight.pdf

[13] BENK,Radomír. *Možnosti technologie Adobe AIR*.2010. Bakalářská práce. VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

[14] NOHA,Pavol. *Webové rozhraní pro správu procesů výkonnostního testování SIP infrastruktury*.2013. Diplomová práce. VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA.

[15] VOZŇÁK, Miroslav. *Voice over IP*. Ostrava, 2012. Skripta. VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

[16] *Sipml5*. [online]. [cit. 2015-05-03]. Dostupné z: https://code.google.com/p/sipml5/

[17] *Sipml5 live demo*. [online]. [cit. 2015-05-03]. Dostupné z: http://sipml5.org/

[18] LORENTO, Salvatore a Simon ROMANO. *Real-Time Communication with WebRTC Peer-to-Peer in the Browser*. O'Reilly Media, 2014. ISBN 978-1-44937-187-6.

[19] DIOP, Mamadou. *Technical guide*. [online]. 2013 [cit. 2015-05-04]. Dostupné z: http://webrtc2sip.org/technical-guide-1.0.pdf

[20] FORMÁNEK, Patrik. *Installing WebRTC2SIP gateway*. [online]. 2014.

[cit.2015-05-03]. Dostupné z: http://nil.uniza.sk/sip/installing-webrtc2sip-gateway-tutorial

[21] *WebRTC: Sipml5 with Asterisk 13 on Centos 6.6*. [online]. 2015 [cit. 2015-05-03]. Dostupné z: https://kunjans.wordpress.com/2015/01/09/web-sip-client-sipml5-with-asterisk-13-on-centos-6-6/

[22] KOLÁČEK, Jan a Kateřina KONEČNÁ. *Jak pracovat s MATLABem*. Dostupné z: https://www.math.muni.cz/~kolacek/vyuka/vypsyst/navod.pdf

Seznam příloh

Součástí DP je CD.

Adresářová struktura přiloženého CD:

- matlab
- tcpdump

Příloha A: *Skript pro analýzu audio paketů*

```
%výpočet teoretických hodnot
Cr=64;
Ps=160*8;
deltat=Ps/Cr;
Bv=((78*8)+Ps)/deltat;

%import vstupních dat z excelu
data1=ddeinit('excel','50 to 51 g711aSRTP.xls');% typ
souboru a jeho název
t=ddereq(data1,'r2c1:r100c1');% určení sloupce a řádků ve
formátu:
l=ddereq(data1,'r2c2:r100c2'); %počátek(r-řádek, c-
sloupec):konec výběru

%počáteční nastavení parametrů
i=1;
lenght=0;
%cyklus pro součet velikosti jednotlivych paketu
while t(i)<=1.5 % dokud čas nebude blízko 1,5 ms
  lenght=lenght+l(i);% sčítame jednotlive delky paketu
  i = i+1; %posouváme se dál ve vektoru
end

B1=(lenght/t(i+1))/1000; % upřesnění objemu dat na kbit/s


%import vstupních dat z excelu
data2=ddeinit('excel','51 to 50 g711aSRTP.xls');
t=ddereq(data2,'r2c1:r100c1');
l=ddereq(data2,'r2c2:r100c2');

%počáteční nastavení parametrů
i=1;
lenght=0;

while t(i)<=1.5 % dokud cas nebude blizko 1 ms
  lenght=lenght+l(i);% scitame jednotlive delky paketu
  i = i+1 %posouvame se dal ve vektoru
end

B2=(lenght/t(i+1))/1000; % upresneni objemu dat na kbit/s
```

```
disp('Vypocitany objem dat je[kbit/s]:')
disp(Bv)
disp('Namereny objem dat je [kbit/s]:')
disp(B1)
disp(B2)

B=(B1+B2)/2

Y=[Bv,B1,B2];%vytvoření vektoru hodnot
X = cell(1,3);%vytvoření popisku osy
X{1}='Vypočítaná hodnota'; X{2}='Naměřená hodnota 1';
X{3}='Naměřená hodnota 2';
bar(Y,'b')% funkce pro vykreslení grafu
set(gca,'xticklabel', X)% přiřazení legendy k ose x

title('Porovnání výsledků UDP paketů');%Titulek grafu
ylabel('[kbit/s]')% legenda osy y
```

Příloha B: *Skript pro analýzu video paketů*

```
%import vstupních dat z excelu
data1=ddeinit('excel','V1.1.xls');

t=ddereq(data1,'r2c1:r2295c1');
p=ddereq(data1,'r2c2:r2295c2');
l=ddereq(data1,'r2c5:r2295c5');

%počáteční nastavení parametrů
i=1;
lenght=0;
time_c=0;
while t(i+1)~=0 % označení konce cyklu
  if p(i)==1%když je protokol UDP
      lenght=lenght+l(i);
      time=t(i+1)-t(i);%výpočet doby přenosu paketu
      time_c=time_c+time;%výpočet celkového času
  end
  i = i+1;
end

B1=(lenght/time_c)/1000

%import vstupních dat z excelu
data2=ddeinit('excel','V1.2.xls');
```

```
t=ddereq(data2,'r2c1:r2299c1');
p=ddereq(data2,'r2c2:r2299c2');
l=ddereq(data2,'r2c5:r2299c5');

%počáteční nastavení parametrů
i=1;
lenght=0;
time_c=0;
while t(i+1)~=0 % označení konce cyklu
  if p(i)==1%když je protokol UDP
      lenght=lenght+l(i);
      time=t(i+1)-t(i);%výpočet doby přenosu paketu
      time_c=time_c+time;%výpočet celkového času
  end
  i = i+1;
end

B2=(lenght/time_c)/1000

%import vstupních dat z excelu
data3=ddeinit('excel','V2.1.xls');

t=ddereq(data3,'r2c1:r2299c1');
p=ddereq(data3,'r2c2:r2299c2');
l=ddereq(data3,'r2c5:r2299c5');

%počáteční nastavení parametrů
i=1;
lenght=0;
time_c=0;
while t(i+1)~=0 % označení konce cyklu
  if p(i)==1%když je protokol UDP
      lenght=lenght+l(i);
      time=t(i+1)-t(i);%výpočet doby přenosu paketu
      time_c=time_c+time;%výpočet celkového času
  end
  i = i+1;
end

B3=(lenght/time_c)/1000

%import vstupních dat z excelu
data4=ddeinit('excel','V2.2.xls');
```

```
t=ddereq(data4,'r2c1:r2299c1');
p=ddereq(data4,'r2c2:r2299c2');
l=ddereq(data4,'r2c5:r2299c5');

%počáteční nastavení parametrů
i=1;
lenght=0;
time_c=0;
while t(i+1)~=0 % označení konce cyklu
  if p(i)==1%když je protokol UDP
      lenght=lenght+l(i);
      time=t(i+1)-t(i);%výpočet doby přenosu paketu
      time_c=time_c+time;%výpočet celkového času
  end
  i = i+1;
end

B4=(lenght/time_c)/1000

%import vstupních dat z excelu
data5=ddeinit('excel','V3.1.xls');

t=ddereq(data5,'r2c1:r2299c1');
p=ddereq(data5,'r2c2:r2299c2');
l=ddereq(data5,'r2c5:r2299c5');

%počáteční nastavení parametrů
i=1;
lenght=0;
time_c=0;
while t(i+1)~=0 % označení konce cyklu
  if p(i)==1%když je protokol UDP
      lenght=lenght+l(i);
      time=t(i+1)-t(i);%výpočet doby přenosu paketu
      time_c=time_c+time;%výpočet celkového času
  end
  i = i+1;
end

B5=(lenght/time_c)/1000

%import vstupních dat z excelu
data6=ddeinit('excel','V3.2.xls');

t=ddereq(data6,'r2c1:r2299c1');
```

```
p=ddereq(data6,'r2c2:r2299c2');
l=ddereq(data6,'r2c5:r2299c5');

%počáteční nastavení parametrů
i=1;
lenght=0;
time_c=0;
while t(i+1)~=0 % označení konce cyklu
  if p(i)==1%když je protokol UDP
      lenght=lenght+l(i);
      time=t(i+1)-t(i);%výpočet doby přenosu paketu
      time_c=time_c+time;%výpočet celkového času
  end
  i = i+1;
end

B6=(lenght/time_c)/1000

B=(B1+B2+B3+B4+B5+B6)/6

Y=[B1,B2,B3,B4,B5,B6,B]
X = cell(1,7);
X{1}='1'; X{2}='2';X{3}='3';
X{4}='4'; X{5}='5';X{6}='6';
X{7}='Průměr';
bar(Y,'b')
set(gca,'xticklabel', X)

title('Porovnání výsledků UDP video paketů');    %Titulek
ylabel('[kbit/s]')
xlabel('Měření č.')
```

Příloha C: *Skript pro vyhodnocení zadaných parametrů*

```
clc; clear all; close all

%vstupni parametry
p_audio = input('Zadejte počet audio hovorů: ');
p_video = input('Zadejte počet video hovorů: ');
param = input('Vyberte protokol pro přenos (RTP=1, SRTP=2,
UDP=3, všechny=4): ');

%hodnoty pro výpočet
a_rtp=90.3265;
a_srtp=94.3235;
```

```
a_udp=84.7145;
v_rtp=894.6884;
v_srtp=898.1598;
v_udp=874.9270;

c_rtp=(p_audio*a_rtp)+(p_video*v_rtp);
c_srtp=(p_audio*a_srtp)+(p_video*v_srtp);
c_udp=(p_audio*a_udp)+(p_video*v_udp);

switch param
    case 1
        fprintf('Zvolili jste %d audio hovorů a %d
videohovorů\n', p_audio, p_video);
        disp('Zvolený kodek pro přenos je RTP');
        fprintf('Celkové nároky na přenos jsou: %f kbit/s
\n',c_rtp);
    case 2
        fprintf('Zvolili jste %d audio hovorů a %d
videohovorů\n', p_audio, p_video);
        disp('Zvolený kodek pro přenos je SRTP');
        fprintf('Celkové nároky na přenos jsou: %f kbit/s
\n',c_srtp);
    case 3
        fprintf('Zvolili jste %d audio hovorů a %d
videohovorů\n', p_audio, p_video);
        disp('Zvolený kodek pro přenos je UDP');
        fprintf('Celkové nároky na přenos jsou: %f kbit/s
\n',c_udp);
    case 4
        fprintf('Zvolili jste %d audio hovorů a %d
videohovorů\n', p_audio, p_video);
        disp('Zvolili jste všechny typy kodeků');
        fprintf('Celkové nároky na přenos pomocí RTP jsou: %f
kbit/s \n',c_rtp);
        fprintf('Celkové nároky na přenos pomocí SRTP jsou: %f
kbit/s \n',c_srtp);
        fprintf('Celkové nároky na přenos pomocí UDP jsou: %f
kbit/s \n',c_udp);
end
```